**TECHNICKÁ UNIVERZITA V LIBERCI**
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: N2612 – Elektrotechnika a informatika

Studijní obor: 1802T007 – Informační technologie

# Program pro kontrolu webových odkazů

# Web links checker

**Diplomová práce**

Autor: **Bc. Vojtěch Domorád**

Vedoucí práce: doc. RNDr. Pavel Satrapa, Ph.D.

**V Liberci 15. 8. 2012**

TECHNICKÁ UNIVERZITA V LIBERCI
**Fakulta mechatroniky, informatiky a mezioborových studií**
Akademický rok: **2011/2012**

# ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

| | |
|---|---|
| Jméno a příjmení: | **Bc. Vojtěch Domorád** |
| Osobní číslo: | **M09000210** |
| Studijní program: | **N2612 Elektrotechnika a informatika** |
| Studijní obor: | **Informační technologie** |
| Název tématu: | **Program pro kontrolu webových odkazů** |
| Zadávající katedra: | **Ústav nových technologií a aplikované informatiky** |

Zásady pro vypracování:

1. Proveďte rešerši existujících nástrojů pro kontrolu webových odkazů, analyzujte jejich přednosti a nedostatky.
2. Navrhněte vlastní program pro kontrolu odkazů na WWW stránkách.
3. Hlavní pozornost při návrhu věnujte uživatelskému rozhraní při zadávání cíle kontroly a práci se zjištěnými výsledky.
4. Program implementujte, otestujte a popište.

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **cca 60 stran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

**[1] Eubanks, Brian. Java na maximum. Computer Press 2006. ISBN 8025111113.**
**[2] Herout, Pavel. Java grafické uživatelské rozhraní a čeština. Kopp 2001. ISBN 80-7232-237-0.**
**[3] Harold, Elliotte Rusty. Java Network Programming. O'Reilly Media 2004, ISBN 0-596-00721-3.**
**[4] Fielding, Roy T. Hypertext Transfer Protocol – HTTP/1.1. IETF 1999. RFC 2616.**

Vedoucí diplomové práce: **doc. RNDr. Pavel Satrapa, Ph.D.**
Ústav nových technologií a aplikované informatiky

Datum zadání diplomové práce: **14. října 2011**
Termín odevzdání diplomové práce: **18. května 2012**

prof. Ing. Václav Kopecký, CSc.
děkan

Technická univerzita v Liberci - 14 -

prof. Dr. Ing. Jiří Maryška, CSc.
vedoucí ústavu

V Liberci dne 14. října 2011

# Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum

Podpis

# Poděkování

Na tomto místě bych rád poděkoval doc. RNDr. Pavlovi Satrapovi, Ph.D. za odborné vedení při vypracování této diplomové práce. Zároveň bych chtěl poděkovat své rodině a přátelům, že mě po celou dobu studia podporovali, a své přítelkyni za trpělivost.

# Abstrakt

Tato diplomová práce je věnována problematice odkazů na webových stránkách jako jedné z klíčových podmínek vyhledávání, sdílení a přenášení dat. Teoretická část popisuje základní historické aspekty vývoje webových stránek, následně se věnuje protokolům pro výměnu hypertextových dokumentů, stavovým kódům a adresám webových stránek. V praktické části se autor zaměřil na analýzu a srovnání v současnosti nejvíce užívaných nástrojů pro kontrolu webových odkazů a v návaznosti na ni vytvořil vlastní program pro kontrolu odkazů na webových stránkách, který umožňuje uživateli rychlou a přehlednou orientaci v množství odkazů na webových stránkách.

***Klíčová slova:*** HTML dokument, Java, stavový kód HTTP, URL

# Abstract

This thesis is devoted to links on websites as one of the key search terms, share and data transfer. The theoretical part describes basic historical aspects of website development. Then it focuses on protocols for hypertext documents exchange, status codes and website addresses. In practical part, the author focuses on the analysis and comparison of the most current used tools to check website links. In addition, the author created his own program to check links on web websites, which allows users quick orientation in number of links on a website.

***Key words:*** HTML document, Java, HTTP status code, URL

# Obsah

Prohlášení ... 3

Poděkování ... 4

Abstrakt ... 5

Abstract ... 6

Seznam obrázků ... 9

Seznam tabulek ... 9

Úvod ... 10

1 Teoretická část ... 12

1.1 Historie a vývoj webu ... 12

1.2 Protokol HTTP (Hypertext Transfer Protocol) ... 16

1.2.1 Vývoj HTTP protokolu ... 16

1.2.2 HTTP komunikace ... 19

1.3 Stavové kódy HTTP ... 20

1.4 Adresy ve WWW ... 24

1.4.1 URI, URL, URN ... 24

1.4.2 URI syntax ... 26

1.4.3 Schémata ... 28

1.4.4 Absolutní a relativní adresy ... 31

2 Rešerše současných nástrojů ... 33

2.1 Online nástroje ... 33

2.1.1 W3C Link Checker ... 33

2.2 Desktopové aplikace ... 35

2.2.1 Xenu's Link Sleuth ... 35

2.2.2 LinkChecker ... 37

2.2.3 Web Link Validator ... 38

2.2.4 HTML Link Validator ... 40

3 Analýza a návrh aplikace ..... 43
3.1 Proč se zabývat tvorbou dalšího softwaru ..... 43
3.2 Analýza zadaného úkolu ..... 43
4 Vývoj aplikace ..... 46
4.1 Programovací jazyk a vývojové prostředí ..... 46
4.2 Strukturování projektu ..... 47
5 Jádro aplikace ..... 49
5.1 Zpracování odkazů ..... 49
5.2 Třída LinkReader ..... 49
5.2.1 Získávání odkazů ..... 52
5.3 Třída LinkInformator ..... 53
6 Grafické uživatelské prostředí ..... 54
6.1 Web Links Checker ..... 54
6.2 Panel nabídek ..... 55
6.3 Panel s výsledky ..... 57
6.4 Předvolby (Preferences) ..... 57
6.5 Spuštění testování ..... 59
6.6 Filtrování odkazů ..... 59
6.7 Chybné odkazy ..... 61
Závěr ..... 62
PŘÍLOHA A ..... 65
PŘÍLOHA B ..... 68

# Seznam obrázků

Obrázek 1: Chronologický vývoj HTML (zdroj: http://appleinsider.com) ..... 15
Obrázek 2: Error 404 - File Not Found ..... 24
Obrázek 3: Diagram kategorií schématu URI (zdroj: http://cs.wikipedia.org) ..... 25
Obrázek 4: URI adresa a její rozdělení na části ..... 28
Obrázek 5: W3C Link Checker ..... 35
Obrázek 6: Xenu's Link Sleuth ..... 36
Obrázek 7: LinkChecker – rozhraní příkazového řádku ..... 38
Obrázek 8: LinkChecker – grafické uživatelské rozhraní ..... 38
Obrázek 9: Web Link Validator ..... 40
Obrázek 10: HTML Link Validator ..... 42
Obrázek 11: Schéma návrhu ..... 43
Obrázek 12: Vývojový proces navrhované aplikace ..... 44
Obrázek 13: Základní struktura projektu ..... 48
Obrázek 14: Hlavní okno aplikace a její části ..... 55
Obrázek 15: Okno nastavení předvoleb ..... 58
Obrázek 16: Okno nastavení filtrů ..... 60
Obrázek 17: Okno filtrování podle uživatelsky definovaných kódů ..... 60
Obrázek 18: Zobrazení chybných odkazů v jednom HTML dokumentu ..... 61

# Seznam tabulek

Tabulka 1: Metody požadavků HTTP ..... 18
Tabulka 2: Kategorie stavových hlášení ..... 20
Tabulka 3: Stavové kódy a hlášení ..... 21
Tabulka 4: Ukázky skládání URL ..... 32
Tabulka 5: Shrnutí vlastností programu W3C Link Checker ..... 34
Tabulka 6: Shrnutí vlastností programu Xenu's Link Sleuth ..... 36
Tabulka 7: Shrnutí vlastností programu LinkChecker ..... 37
Tabulka 8: Shrnutí vlastností programu Web Link Validator ..... 39
Tabulka 9: HTML Link Validator ..... 41
Tabulka 10: Seznam vyhledávaných tagů ..... 52

# Úvod

V dnešní době se lidstvo neustále snaží získávat a zpracovávat co nejvíce informací o světě kolem sebe a také zkracovat a zrychlovat přenos informací - dat mezi jednotlivými uživateli. Možnost propojení počítačů přinesla velký rozmach získávání informací a možnosti elektronické komunikace. Původně americký armádní projekt (Arpanet) propojující elektronicky několik samostatných velitelských center strategické protivzdušné obrany byl později předán do akademického světa, kde se k němu připojovaly další a další počítače a sítě. Vznikla soustava sítí, které jsou mezi sebou propojeny (Internet) a jež v současné době obklopují celou zeměkouli. Společnost tohoto tisíciletí bývá proto právem označována jako informační společnost.

Informační technologie dnes nabízejí možnost využití informací jak v pracovním tak v soukromém životě každého jedince. Dnes není proto problém získat informace, ale zpracovat je, utřídit a přehledně ukládat. Lidé jsou zavaleni zprávami a informacemi různého charakteru. Velmi mnoho informací lze předat pomocí internetu, a to například prostřednictvím webových stránek, jako jedné z mnoha jeho služeb. Velké nároky jsou zde kladeny především na webové stránky a jejich autory, kteří je vytvářejí. Informace, které jsou na nich umístěny, a to ať už jde o text, obrázek, zvuk, video či hypertextový odkaz, musí být funkční a uživatel, který chce dané stránky navštívit, by se k nim měl dostat.

Předložená diplomová práce se zabývá problematikou odkazů na webových stránkách, jako jedné z klíčových podmínek fungování webových stránek. Cílem práce je analýza existujících nástrojů pro kontrolu webových odkazů a následně tvorba vlastního programu pro kontrolu odkazů na webových stránkách.

V teoretické části jsou popsány základní aspekty webových stránek, a to od historie jejich vzniku, až po současnost. Dále je zde popsán protokol pro výměnu hypertextových dokumentů mezi klientem a webovým serverem (HTTP) a stavové kódy, jež jsou součástí všech požadavků na dotyčný server. Další kapitola popisuje základní adresy na webových stránkách, kde se autor věnuje možným tvarům adres a schématům určující typ zdroje a jejich syntaxe. Na závěr této kapitoly je popsán rozdíl mezi absolutní a relativní adresou.

Na teoretickou část navazuje část praktická, kde je nejprve popsáno základní dělení nástrojů pro kontrolu webových odkazů. Následuje velice podrobná rešerše současných

a dostupných nástrojů pro kontrolu webových odkazů, kde jsou uvedeny jejich výhody a nevýhody. Na základě analýzy dostupných programů autor této práce navrhl vlastní program pro kontrolu odkazů na webových stránkách, který je popsán v následující kapitole. Autor se zde věnuje především stavbě jádra aplikace, která má na starosti získávání, testování a následné ukládání odkazů. Velká pozornost je také věnována přehlednému grafickému uživatelskému (rozhraní/prostředí) a zpracování výsledků, s nimiž souvisí testování aplikace formou kontroly odkazů na různých webových sítích, filtraci nalezených odkazů a následnému zobrazení chybných odkazů. Příloha práce pak obsahuje manuál k vytvořené aplikaci pro kontrolu odkazů, kde je popsána základní práce s programem včetně praktických návodů.

# 1 Teoretická část

První kapitola je věnována vymezení pojmů spjaté s tématem a problematikou této práce. Čtenář se seznámí se vznikem a vývoje Hypertext Markup Language[1] (HTML), strukturou webových odkazů a typy návratových kódů při práci s odkazy.

## 1.1 Historie a vývoj webu

Tato kapitola popisuje ve stručnosti historii a vývoj HTML jazyka. Cílem je ukázat, jak byl tento jazyk, se kterým se setkáváme každý den, vyvíjen od prvního konceptu napsaného Timem Berners-Leem na začátku 90. let 20. století do současné podoby. Než byl jazyk HTML standardizován, prošel velice obtížnou cestou. Byl diskutován různými skupinami lidí od softwarových inženýrů po ministry z Dolní Komory Spojeného království.

**World Wide Web** je výsledkem mnohaletého evolučního vývoje.

Počátky jazyka HTML spadají už do 80. let minulého století, kdy Tim Berners-Lee pracoval v ženevských laboratořích pro Evropský institut pro jaderný výzkum (CERN). V laboratořích CERNu bylo v tu dobu do projektu výzkumu částicové fyziky zapojeno kolem 10 tisíc vědců z různých institucí z celého světa. V roce 1989 Berners-Lee pracoval v sekci výpočetní techniky, když přišel s konceptem umožňujícím výzkumníkům ze vzdálených sítí na celém světě organizovat a sdílet informace mezi sebou. Nechtěl však jenom jednoduché sdílení souborů o výzkumu, které by se daly stáhnout ze vzdáleného počítače. Navrhoval, aby bylo možné odkazovat na text v souborech samotných.

Tim Berners-Lee se snažil vytvořit „zastřešující“ službu umožňující odkazování v rámci dokumentů. WWW umožňovalo odkazování na informace dokonce i v rámci různých síťových služeb.

V roce 1991 přišel Berners-Lee s prohlížečem, zprovoznil svůj web v CERNu a vydal první veřejně dostupnou definici jazyka HTML, která byla popsána v dokumentu nazvaném „HTML Tags“. Bylo v ní popsáno 18 jednoduchých konstrukčních prvků jazyka HTML. Hlavní předností byla možnost rozčlenit text do několika logických úrovní, použít několik druhů zvýraznění textu a přidat do textu odkazy a obrázky.

[1]Hypertext Markup Language je jeden z jazyků sloužících pro vytváření dokumentů v systému World Wide Web.

Od této chvíle požadavky uživatelů webu vzrůstaly. Proto vývojáři webových prohlížečů obohacovali své produkty, především jazyk HTML o další prvky. Pro zachování kompatibility mezi jednotlivými modifikacemi HTML a snahy pro definování HTML jako jednoho z validních SGML[2] jazyků (Standard Generalized Markup Language) byl v roce 1993 vytvořen návrh skrze standardizační autoritu Internet Engineering Task Force[3] (IETF) nazvaný „Hypertext Markup Language (HTML) - Internet-Draft". IETF následně pod svojí záštitou vytvořila speciální pracovní skupinu „Integration of Internet Information Resources". Tato skupina měla za úkol sjednotit všechny verze HTML. V listopadu vyšla 1995 konečná specifikace Hypertext Markup Language 2.0. Publikována byla v dokumentu Request for Comments[4] pod číslem 1866. Tento dokument je považován za standard, ze kterého by měly vycházet všechny budoucí implementace.

Protože už v této době bylo vidět velký potenciál HTML jazyka, odešel Tim Berners-Lee z technologického institutu v CERNu a v říjnu roku 1994 založil World Wide Web konsorcium (W3C). Toto konsorcium bylo založeno na Massachusettském technologickém institutu za pomoci Evropské komise a Agentury pro výzkum pokročilých obranných projektů (DARPA) za účelem vývoje a sjednocení všech dostupných modifikovaných verzí jazyka HTML. Další vývoj HTML pracovní skupinou pod záštitou IETF byl proto z důvodu konkurenčních zájmů zastaven.

Vývoj standardů webu v té době již řídilo konsorcium W3C, jehož členy se staly a stále ještě jsou přední softwarové firmy, mezi něž patří například IBM Corporation, Microsoft, Nescape. Tyto společnosti přispívaly k rozšíření HTML všemi dostupnými prostředky – jak v podobě návrhů, tak i finančních darů. V dubnu ještě tentýž rok byl, co byla vydána specifikace standardu HTML 2.0, byl předložen návrh verze 3.0. Ukázalo se, že návrh HTML 3.0 byl velice složitý a nenašel se nikdo, kdo by dokázal implementovat prohlížeč s plnou podporou všech jeho funkcí.

Od návrhu verze 3.0 bylo nakonec upuštěno a místo něj vznikl návrh HTML verze 3.2, který byl sjednocením všech schopností tehdejších prohlížečů. V lednu 1997 byl vydán jako W3CRecommendation, což je dokument, který prošel náročnou kontrolou a testováním. Dokument s tímto statusem je brán jako standart pro nasazení do praxe a používat by ji měli všichni vývojáři, tak aby byla zajištěna kompatibilita.

[2]SGML je univerzální značkovací jazyk umožňující definovat jiné značkovací jazyky jako své vlastní podmnožiny.
[3]Internet Engineering Task Force (IETF), je volné sdružení lidí vyvíjející internetové technologie.
[4]RFC se používá pro označení řady technických dokumentů popisujících fungování Internetu.

Na konci roku však byl konsorciem W3C vydán dokument HTML verze 4.0, také jako W3C Recommendation. Byly zde nepatrné úpravy oproti předchozí verzi. Jednou z těchto změn je nabídka 3 variant HTML DOCTYPE deklarace – Strict, Transitional a Frameset. Verze Strict nabízí všechny dokumenty HTML, ale nezahrnuje prezentační a zastaralé elementy, např. fonty.

Od té doby se vývoj značkovacího jazyka zastavil. Tedy až v roce 1999, kdy se objevil standard verze 4.01, opravující drobné chyby. V roce 2000, byl dokonce standard HTML 4.01 přijat jako mezinárodní norma ISO/IEC 15445:2000. Později po vydání HTML verze 4.0 vydává konsorcium W3C jazyk XML (eXtensibleMarkupLanguage). Ten se od svého vydání v roce 1998 stal ve světě informačních technologií jednoznačným formátem pro výměnu a ukládání dat. Později v roce 2000 konsorcium W3C vydává specifikaci jazyka XHTML, což je jazyk HTML odvozený od syntaxe XML namísto SGML jako v HTML. Většina značek a způsob zápisu zůstal stejný.

V dalších letech se zdálo XHTML jako správný pro vývoj webových technologií. Konsorcium W3C proto vydává specifikaci jazyka XHTML 1.1 a začíná s vývojem XHTML 2.0. Ale bohužel se ukázalo, že cesta skrze XHTML nebyla tou nejlepší. V té době nejpoužívanější prohlížeč Internet Explorer navíc neuměl XHTML zpracovávat korektně. XHTML 2.0 vypadalo jako ideální řešení přinášející mnohé zajímavé vlastnosti. Bohužel výměnou za ztrátu kompatibility s původním jazykem HTML. Vývojářům, konkrétně společnostem Apple, Mozilla Foundation, Opera Software a Google, navíc začal vadit poměrně pomalý vývoj webových standardů. Proto založili společnou pracovní skupinu WHATWG (Web Hypertext Application Technology Working Group), která měla být odpovědí na tento pomalý vývoj. Na její půdě pak začali připravovat platformu pro webové aplikace běžící v prohlížeči. Kromě poměrně rozsáhlého rozšíření jazyka HTML, specifikace zahrnovala i definici důležitých rozhraní pro využití skriptovacím jazyku Javascript.

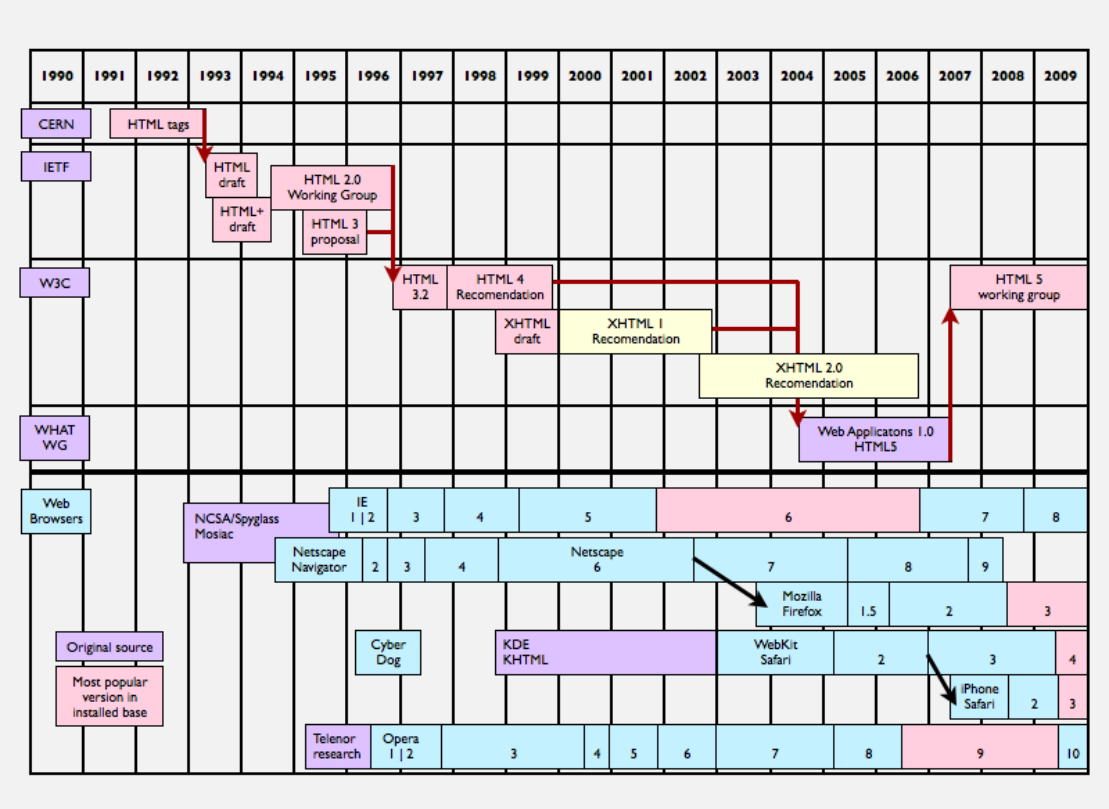
Protože rozdělení vývoje a tedy i kompatibility verzí jazyka HTML bylo nepřípustné, po dlouhých diskuzích uvnitř W3C odhlasováno zastavení vývoje XHTML 2.0 z důvodu ignorace vývojářů webových prohlížečů. V roce 2007 se obě pracovní skupiny spojily a W3C přijalo základ specifikace vytvořené ve WHATWG a na tomto základě začala vznik specifikace HTML5.

HTML5 navazuje na HTML 4.01 a přidává pro vývojáře webových aplikací spoustu užitečných funkcí, kvůli kterým je výhodné tuto nejnovější verzi jazyka používat. Přináší nové, zkrácené a rychlejší zápisy, jako např. zkrácený zápis DOCTYPE, kde už není potřeba uvádět

verzi a DTD specifikaci dokumentu. Autoři navíc dávají důraz na jednoduchost a zároveň účinnost.

V tuto chvíli však ještě specifikace HTML5 není zcela hotová. Některé části HTML5 jsou již odladěné a většina současných nových verzí prohlížečů je již podporuje. Naopak některé části jsou ve fázi vývoje a není jasné, zdali se ve finální specifikaci prosadí nebo ne.

Já osobně si myslím, že HTML 5 v kombinaci s novým CSS je dobrá volba pro novou generaci webů, které opět budou směřovat k jednoduchosti. Velká výhoda je hlavně v tvorbě a přehrávání multimediálním obsahu, který se dá na cílové platformě spouštět bez dalšího podpůrného softwaru jako je např. Adobe Flash[5] či Microsoft Silverlight[6].




**Obrázek 1: Chronologický vývoj HTML (zdroj: http://appleinsider.com)**

[5] Adobe Flash je multimediální platforma, používaná ke vkládání animací, videí, reklam a programů do webových stránek.

[6] Microsoft Silverlight je aplikační platforma vytvořená společností Microsoft, která je určena pro vývoj business a multimediálních aplikací.

## 1.2 Protokol HTTP (Hypertext Transfer Protocol)

World Wide Web, nebo pouze jen Web znamená ve volném překladu „celosvětová pavučina“. Skládá se z ohromného množství webových serverů a milionů klientských systémů, které jsou schopny se s nimi dočasně spojit. Základním pojivem, které drží web pohromadě je sada univerzálních standardů, umožňujících těmto klientům a serverům výměnu informací skrze internet. Tyto definované standardní metody komunikace v síti se nazývají protokoly.

V informatice tento termín označuje konvenci nebo standard, podle kterého probíhá elektronická komunikace a přenos dat v pevně daném formátu mezi dvěma koncovými body (nejčastěji právě mezi serverem a klientským počítačem). Proto byl pro potřeby služby WWW vyvinut protokol pro vzájemnou komunikaci mezi klientem a serverem – protokol HTTP.

**Protokol HTTP** je tzv. aplikační protokol, který pro přenos dat po síti využívá služeb protokolů zajišťující komunikaci na nižší úrovni síťového modelu – protokolu TCP zajišťujícího spojovou službu. Tento protokol vychází z architektury klient-server. Klient, v tomto případě internetový prohlížeč, naváže spojení se serverem a odešle požadavek. Po každém odeslaném požadavku z prohlížeče uživatele, zašle webový server odpověď. Toto spojení se navazuje pouze na dobu nezbytnou pro přenos a po ukončení přenosu je spojení ukončeno. Protokol HTTP je bezstavový, tzn., že nerozlišuje klienty, od nichž chodí požadavky. Pokud nějaký klient odešle požadavek a vzápětí ten stejný klient odešle další požadavek, server nepozná, že jde o stejného klienta.

### 1.2.1 Vývoj HTTP protokolu

V současné době je HTTP protokol k dispozici ve 3 verzích:

- HTTP/0.9
- HTTP/1.0
- HTTP/1.1

První verze protokolu HTTP se datuje do stejné doby jako vznik první verze HTML jazyka – do roku 1991. Za jeho vznikem stojí Tim Berners-Lee. V první verzi HTTP/0.9 existovala pouze jedna metoda GET s jediným parametrem a to názvem požadovaného dokumentu na serveru. Server jako odpověď poslal přímo požadovaný dokument bez hlavičky a případná chybová hlášení vracel ve formě HTML dokumentu. Webové servery, které implementovaly

tuto neoficiální verzi protokolu, odpovídaly na jednoduché dotazy jako např. „GET /welcome.html“. Po přijetí požadavku server odeslal dokument uložený v souboru welcome.html pokud existoval nebo případně chybou hláškou, pokud tomu tak nebylo.

Dalšího vývoje protokolu se ujala pracovní skupina „HTTP Working Group“ v čele s Davidem Raggettem. Při založení skupiny byly pevně stanoveny cíle práce – mezi ně v první řadě patřilo zefektivnění provozu, obohacení o metainformace a provázání s bezpečnostními protokoly.

HTTP/0.9 byla oficiálně nahrazen v květnu 1996, při vydání RFC 1945 - HTTP/1.0. Nejdůležitější věcí přidanou do protokolu verze 1.0 bylo použití hlaviček, popisující přenášená data a dvě nové metody HEAD a POST. Tyto hlavičky říkají prohlížeči, jak má naložit s daty. Nejčastěji používanou hlavičkou na webu je „Content-Type: text/html“. Tato hlavička říká prohlížeči, že data, která následují, tvoří text formátovaný pomocí HTML. Formátovací kódy HTML vložené do textu popisují, jak má prohlížeč stránku zobrazit.

Protokol HTTP verze 1.1, který se stále používá k dnešnímu dni, byl původně popsán v RFC 2068 v lednu 1997, ale v červnu 1999 ji nahradila aktualizovaná definice RFC2616.

Změny v HTTP/1.1 se týkaly především:

- **Udržení spojení (Persistent Connection)**
  Změna se týkala udržení TCP spojení (tzv. keep-alive spojení) mezi klientem a serverem tak, aby bylo možné zpracovat během jednoho spojení více požadavků jdoucích po sobě. Webové stránky se totiž skládají z mnoha komponent – např. multimediálního obsahu (obrázky) a dříve bylo nutné před stažením každé této komponenty navázat nové spojení, čímž docházelo ke zpomalování získávání obsahu dokumentů.
- **Vyjednávání o obsahu (Content Negotiation)**
  Tato vlastnost umožňuje volbu varianty zdroje. Díky tomu je možné nabízet dokument ve více jazykových verzích.
- **Metody požadavků**
  Bylo přidáno několik nových metod požadavků – *Put, Delete, Trace, Options, Connect*, ale s nimi se nesetkáváme tak často.
- **Rozšíření počtu stavových kódu odpovědi**
- **Možnost zasílání odpovědi po částech**

V roce 2000 se objevila ještě specifikace RFC 2774 označovaná jako HTTP/1.2 s rozšířením PEP (Protocol Extension Protocol). Nicméně tato verze návrhu zůstala ve stádiu experimentu a proto není v praxi využívána.

**Tabulka 1: Metody požadavků HTTP**

| **Typ metody** | **Popis metody** |
|---|---|
| *Get* | Načítá prostředek identifikovaný v URL požadavku. |
| *Head* | Shodné s metodou *Get*, ale už nepředává data. Poskytne pouze hlavičky odpovědi o požadovaném cíli (velikost, typ, datum změny, …). |
| *Post* | Odesílá uživatelská data na server. Používá se nejčastěji při odesílání informací zadaných do polí webového formuláře. S předaným objektem se pak zachází podobně jako při metodě GET. Data může odesílat i metoda GET, metoda POST se ale používá pro větší objem dat (víc než 512 bajtů, což je maximální velikost požadavku GET) nebo pokud není vhodné přenášená data zobrazit jako součást URL (data předávaná metodou POST jsou obsažena v HTTP požadavku). |
| *Put* | Umožňuje klientovi odeslat prostředek identifikovaný v URL požadavku na server. Server, pokud bude akceptovat *Put*, ukládá informace, které dostává od klienta. |
| *Delete* | Požaduje po serveru odstranění prostředku, zadaného v URL požadavku. |
| *Trace* | Používá se ke sledování cesty celého dotazu. V těle odpovědi obdrží klient seřazené dotazy všech jednotlivých systémů, kterými požadavek procházel. Je používaná např. webovými programátory, kteří chtějí zjistit, proč jim server vrací například expirovaný dokument apod. |
| *Options* | Dotaz na informaci o možnostech komunikace poskytovaných serverem – typy podporovaných metod. |
| *Connect* | Používá se pro spojení klienta se serverem skrze proxy server na definovaném portu. |

### 1.2.2 HTTP komunikace

Jak již bylo řečeno, tak protokol HTTP je protokolem aplikační úrovně pro distribuované hypermediální informační systémy, postavený na architektuře klient – server. V praxi to tedy znamená, že klient (nejčastěji webový prohlížeč, ale může jím být i vyhledávací robot nebo jiný program) se připojí na webový server a zašle žádost. Webový server tuto žádost zpracuje a pošle na ni klientovi odpověď.

Obecně pokud webový server přijme požadavek od prohlížeče klienta, provede jednu ze dvou akcí. V prvním případě odpoví klientovi zasláním dokumentu (buď staticky nebo dynamicky generovaného programem běžícím na serveru) nebo odmítne na požadavek odpovědět a místo toho odešle číselný stavový kód, charakterizující typ chyby, která nastala.

**Struktura HTTP žádosti a odpovědi:**

**Požadavek od klienta:**

```
HTTP_METODA URL_DOKUMENTU HTTP_VERZE
HLAVIČKY (každá na nový řádek)
prázdný řádek
DATA_Z_FORMULÁŘE (u HTTP-metody POST)
```

**Odpověď serveru:**

```
HTTP_VERZE STAVOVÝ_KÓD STAVOVÉ_HLÁŠENÍ
HLAVIČKY (každá na novém řádku)
prázdný řádek
OBSAH_ODPOVĚDI
```

**Ukázka HTTP komunikace:**

**Požadavek od klienta:**

```
GET /manual/ HTTP/1.1
Host: localhost
```

**Odpověď serveru:**

```
HTTP/1.1 200 OK
Date: Mon, 05 Apr 2004 23:00:38 GMT
Content-Location: index.html.en
```

```
Content-Length: 10051
Content-Type: text/html; charset=ISO-8859-2
Content-Language: en

<html>
<head></head>
<title>stránka</title>
...
</html>
```

## 1.3 Stavové kódy HTTP

Stavové kódy HTTP protokolu jsou používány od verze HTTP/1.0. První řádek v každé zprávě odpovědi je tzv. stavový řádek, sestávající z verze protokolu, číselného stavového kódu a odpovídajícímu text hlášení náležející danému kódu. Stavový kód je 3ciferné číslo a je určen pro zpracování a pochopení strojem. Stavové hlášení je jeho slovní popis určený pro uživatele. Celá dvojice v odpovědi klientovi říká, jak se podařilo splnit jeho požadavek. Klient není povinen zobrazit stavové hlášení. Přesný formát odpovědi je popsán ve specifikaci protokolu HTTP - v dokumentu RFC 2616.

První číslice ve stavovém kódu určuje třídu odpovědi. Zbylé 2 číslice definují přesný typ hlášení. V současné době je definováno 5 tříd chybových hlášení. Tyto třídy jsou vypsány v tabulce 2.

**Tabulka 2: Kategorie stavových hlášení**

| **Kategorie** | **Rozsah stavových kódů** | **Popis** |
|---|---|---|
| Informační | 1xx | Provizorní odpověď podmiňující provedení další akce ze strany žadatele. |
| Úspěch | 2xx | Požadavek byl úspěšně přijat, pochopen a akceptován. |
| Přesměrování | 3xx | Klient musí pro konečné zpracování požadavků vykonat určitou činnost. O této činnosti se uživatel nemusí dovědět. |

| | | |
|---|---|---|
| Chyba na straně klienta | 4xx | Problém na straně klienta – požadavek není korektní nebo nemůže být splněn. |
| Chyba na straně serveru | 5xx | Problém na straně serveru – server neúspěšně splnil zřejmě korektní požadavek. |

V následující tabulce jsou definovány stavové kódy pro specifikaci HTTP/1.1. Stavová hlášení uvedená v této tabulce jsou pouze doporučená – mohou být nahrazeny ekvivalenty v dané oblasti bez ovlivnění protokolu.

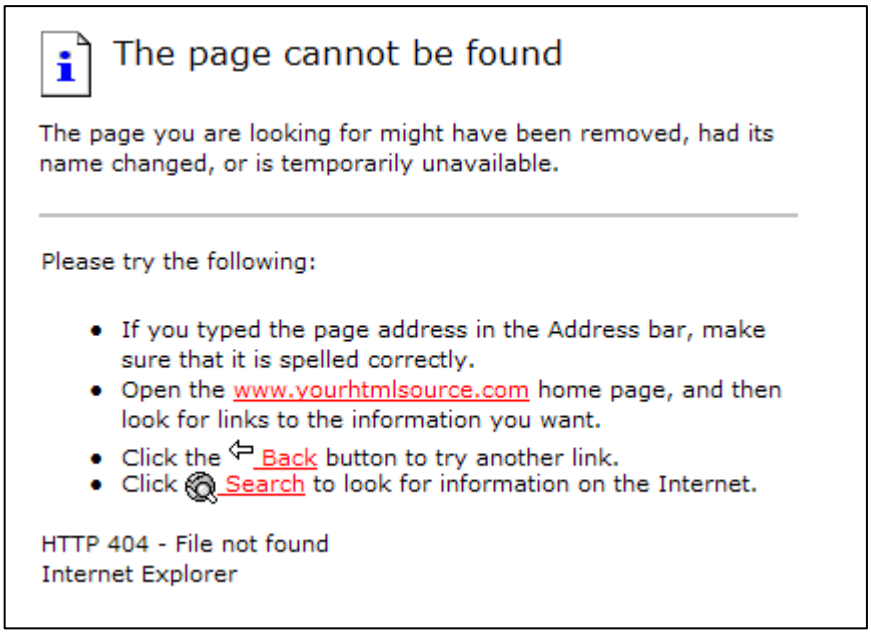
**Tabulka 3: Stavové kódy a hlášení**

| **Stavový kód** | **Stavové hlášení** | **Popis** |
|---|---|---|
| 100 | Continue | Klient může pokračovat v zasílání požadavku. Tato prozatímní odpověď slouží k informování klienta, že inicializační část požadavku byla přijata a dosud nebyla serverem odmítnuta. |
| 101 | Switching Protocols | Server rozumí a souhlasí s požadavkem klienta, pro změnu protokolu. |
| 200 | OK | Žádost byla přijata a úspěšně splněna. |
| 201 | Created | Zdroj identifikovatelný podle URI byl vytvořen. |
| 202 | Accepted | Byl přijat požadavek. Tento požadavek byl správně akceptován, odpovídající činnost se však ještě zatím nemusela provést. |
| 203 | Non-Authoritative Information | Server kladně zpracoval požadavek, ale návratová informace pochází z jiného zdroje. |
| 204 | No Content | Požadavek byl úspěšný, ale jeho výsledkem nejsou žádná data pro klienta. |
| 205 | Reset Content | Server úspěšně zpracoval požadavek, ale nevrací žádnou odpověď a říká klientovi, že smí obnovit původní obsah dokumentu. |
| 206 | Partial Content | Server splnil pouze část požadavku zdroje. Součástí požadavku musí být hlavička Range, která určí požadovaný rozsah. |
| 300 | Multiple Choices | Požadovaný zdroj se dá získat z několika různých míst. V odpovědi se vrací seznam všech možností a chce jej po klientovi specifikovat. |
| 301 | Moved Permanently | Požadovaná adresa URL se trvale přesunula na novou adresu URL. Všechny další odkazy musí použít tuto novou URL. |
| 302 | Found | Požadovaný zdroj byl dočasně umístěn pod jiným URI. Vzhledem k tomu, že přesměrování by mohlo být dočasné, měl by klient i nadále používat požadované URI i pro budoucí |

| | | |
|---|---|---|
| | | požadavky. |
| 303 | See Other | Odpověď na požadavek může být nalezena na jiném URI pomocí metody GET. |
| 304 | Not Modified | Indikuje, že od posledního požadavku se zdrojový dokument nezměnil. Odpověď s tímto kódem nesmí obsahovat tělo. |
| 305 | Use Proxy | Jedná se o bezpečnostní mechanismus. Server prostřednictvím kódu 305 klientovi říká, že k požadavku se musí přistoupit přes proxy v hlavičce „Location“. |
| 306 | (Unused)Switch Proxy | Tento stavový kód byl používán v minulých specifikacích, ale v tuto chvíli se nepoužívá a je rezervován. |
| 307 | Temporary Redirect | Stránka byla dočasně přesunuta na jiné místo. |
| 400 | Bad Request | Server nerozumí požadavku. Příčinou může být chybně formulovaný dotaz nebo chyba v URL adrese. Je potřeba zkontrolovat korektnost požadavku a odeslat poté požadavek znovu. |
| 401 | Unauthorized | Jestliže byl původní požadavek klienta anonymní, musí být nyní autentizován. Pokud už požadavek byl autentizován, pak byl přístup odepřen. |
| 402 | Payment Required | Rezervováno pro budoucí užití. Původní záměr byl využít tento stavový kód v internetových mikroplatebních službách, nicméně k tomu prozatím nedošlo. |
| 403 | Forbidden | Server nemůže odpovědět na požadavek - nemá potřebné oprávnění. |
| 404 | Not Found | Server nenašel požadovanou adresu URL. Tento chybový kód je nejčastější. Příčinou může buď překlep v zápisu URL, nebo neexistence adresy URL. |
| 405 | Method Not Allowed | Použitá metoda není přípustná pro dosažení požadovaného objektu. |
| 406 | Not Acceptable | Požadovaný objekt není k dispozici ve formátu podporovaném klientem. |
| 407 | Proxy Authentication Required | Před obsloužením požadavku musí být tento požadavek autentifikován proxy serverem. |
| 408 | Request Timeout | Klient nedokončil odesílání požadavku v časovém limitu. |
| 409 | Conflict | Požadavek nemohl být splněn z důvodu konfliktu s aktuálním stavem zdroje. |
| 410 | Gone | Požadovaná stránka již není a v budoucnu nebude nadále dostupná. |
| 411 | Length Required | Server neakceptoval požadavek, protože hlavička "Content-Lenght" není definována. Tento stav se obvykle používá jen pro metody HTTP, jejichž výsledkem je umístění dat na webový server, nikoliv čtení z něj. |
| 412 | Precondition Failed | Podmínka, která je zadaná v požadavku, byla serverem vyhodnocená jako chybná. |

| | | |
|---|---|---|
| 413 | Request Entity Too Large | Server neakceptoval požadavek, protože požadované množství je příliš velké. |
| 414 | Request-URI Too Long | Požadavek nebyl akceptován serverem. Chyba se objeví, je-li "POST" požadavek překonvertován na požadavek "GET" s dlouhou dotazovací informací. |
| 415 | Unsupported Media Type | Server neakceptoval požadavek, protože typ média není podporován. |
| 416 | Requested Range Not Satisfiable | Je-li v požadavku hlavička „Range“ vyplněna rozmezím hodnot, které nevyhovují rozsahu hodnot aktuálně vybraného zdroje, může server vrátit tuto chybu. |
| 417 | Expectation Failed | Předpoklad zadaný v hodnotě hlavičky požadavku „Expect“ nemůže server dosáhnout. |
| 500 | Internal Server Error | Na serveru došlo k neočekávané chybě. |
| 501 | Not Implemented | Tento typ požadavku není na serveru podporován. |
| 502 | Bad Gateway | Proxy server nebo brána obdržely od dalšího serveru neplatnou odpověď. |
| 503 | Service Unavailable | Server dočasně nemůže nebo nechce zpracovat požadavek. Většinou když je přetížený nebo se provádí údržba. |
| 504 | Gateway Timeout | Server v pozici brány případně proxy neobdržel včas odpověď od nadřazeného serveru. |
| 505 | HTTP Version Not Supported | Server nepodporuje verzi HTTP v daném požadavku. |

V tuto chvíli se zvažuje ještě použití chybového hlášení 451 – Unavailable For Legal Reasons. O tomto chybovém hlášení se začalo mluvit hlavně kvůli možnosti blokování serveru The Pirate Bay ve Velké Británii. Chybový stavový kód by měl server operátora odeslat prohlížeči uživateli v případě, kdy dojde k nedostupnosti cílového serveru z důvodu porušení zákona. Na stránce se pak může namísto nic neříkající chyby zobrazit i dodatečná informace s podrobnostmi a odůvodněním.




**Obrázek 2: Error 404 - File Not Found**

## 1.4 Adresy ve WWW

Do sítě internetu je zapojeno obrovské množství serverů poskytujících různé služby. Nicméně abychom službu mohli využívat, musíme vědět, na jaké adrese se daný server se službou nalézá. Je tedy na místě, aby existoval univerzální tvar adresy, který by umožnil obsáhnout celou paletu služeb či zdroj informací v internetové síti. Zároveň aby jedna adresa ukazovala právě na jednu službu či zdroj v Internetu. A poslední kritérium této adresy – měla by být snadno čitelná pro člověka i pro počítač. Proto v rámci rozvoje WWW vznikl způsob jak identifikovat jednotlivé zdroje v Internetu.

### 1.4.1 URI, URL, URN

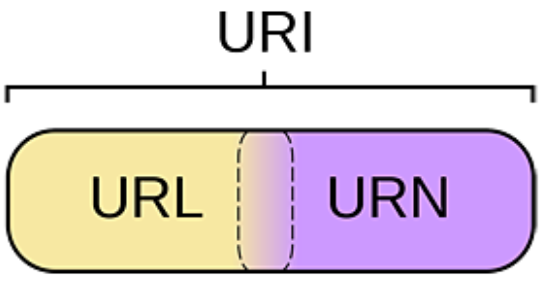
Na internetu se najde mnoho diskuzí a článků na vysvětlení těchto termínů. A všechny říkají přibližně to samé – existuje několik typů jednotných identifikátorů zdrojů. Jednotný identifikátor zdroje (Unified Resource Locator, URI) je z těchto 3 nejobecnější. Dalšími identifikátory jsou - identifikátor jména (Unified Resource Name, URN), který je znám méně, a identifikátor umístění zdroje (Unified Resource Locator, URL), se kterým se setkal určitě každý. URL a URN jsou podtypy URI.

**URI** může popisovat zdroj jak z hlediska identity, tak z hlediska toho, kde je možno zdroj nalézt. Současně může určovat ale i obojí – přesnou identitu zdroje i jak je možno ho

dosáhnout. Pokud bychom si měli ukázat rozdíly na konkrétním příkladu např. autora této práce, tak URN určuje jméno autora, zatímco URL určuje, kde můžeme autora nalézt – tzn. trvalé bydliště. Dalším názorným příkladem je ISBN kód, který je unikátní pro každou knihu. ISBN 0-486-27557-4 (urn:isbn:0-486-27557-4) určuje konkrétní vydání Shakespearovy divadelní hry Romeo a Julie. K dosažení této knihy však musíme znát její umístění: URL adresu. URL v tomto případě by mohla identifikovat elektronickou knihu uloženou na místním pevném disku. Adresa pro tuto knihu by v systému založeném na platformě Windows, mohla vypadat následovně: `file:///C:/Users/username/Documents/RomeoAJulie.pdf`.

**URN** je obecný koncept, představující jednoznačné jméno konkrétního zdroje. Podle tohoto jedinečného jména by měl klient zdroj rozpoznat a obstarat si jej. Z URN se klient dozví, co je daný cíl zač. Jediným problémem je, že dosud nebyl k URN vymyšlen vhodný mechanismus, jak dotyčný dokument najít a získat. Tzn., že v současné době zůstává URN pouze teoretickým konceptem a na webových stránkách se proto využívají lokátory.

**URL** popisuje konkrétní umístění daného zdroje. Kromě identifikace umístění zdroje určuje prostředky, které zastupují zdroj: buď skrze popis přímého přístupového mechanismu, nebo skrze síťové umístění. Zkráceně řečeno, URL obsahuje veškeré informace potřebné pro získání konkrétního umístění dotyčného zdroje – tzn. jakou síťovou službu použít, na který server se obrátit a co po něm chtít. Například URL adresa `http://cs.wikipedia.org/wiki/Hlavní strana`, identifikuje zdroj hlavní stránky české verze Wikipedie, jejíž zastoupení v podobě hlavní stránky je dostupné skrze Hypertext Transfer Protokol ze sítě, jejíž hostitel domény je „`cs.wikipedia.org`".




**Obrázek 3: Diagram kategorií schématu URI (zdroj: http://cs.wikipedia.org)**

### 1.4.2 URI syntax

Původně měl nadpis této kapitoly znít URL syntax. Nicméně všechny dokumenty napsané k URL syntaxi vycházejí z popisu URI syntaxe, a pokud budu citovat část dokumentu RFC 3986, tak v něm je uvedeno, že: „Budoucí specifikace a související dokumentace by měly používat obecný termín "URI" spíše než více restriktivní podmínky "URL" a "URI".“

Obecný tvar URI je definován jako řetězec znaků, skládající se ze čtyř hlavní částí:

```
<schéma>:<hierarchická_část>[ ?<dotaz> ][ #<fragment> ]
```

**Schéma** určuje typ zdroje, na který adresa ukazuje. Skládá se z posloupnosti několika znaků, začínající písmenem a je následován libovolnou kombinací písmen, číslic, znaky: plus, tečka nebo pomlčka. Typ schématu je velice často a nesprávně označován jako protokol, i když původně byl určen k použití s konkrétními protokoly a měl s uvedeným protokolem stejné jméno. Například, typ schématu http, je obecně používán pro interakci s webovými zdroji pomocí Hypertext Transfer Protokolu. V dnešní době, jsou však URI s tímto typem schématu používány i pro jiné účely – např. jako identifikátory RDF[7] zdrojů a ten jako takový se k tomuto protokolu nevztahuje. Kromě toho, některé typy schémat nejsou spojeny s žádnými protokoly jako např. schéma „file“. Navíc u mnoha dalších typů schémat se neshoduje název schématu s typem používaného protokolu, např. „news“.

**Hierarchická část** URI slouží k identifikaci zdroje a zpravidla začíná dvěma dopřednými lomítky. Hierarchickou část je možné dále rozdělit na část autorizační a cestu zdrojového dokumentu.

- **Autorizační část** obsahuje uživatelské informace (jméno uživatele a heslo, kterým se daný uživatel připojuje k nějaké službě, jsou odděleny dvojtečkou a ukončeny znakem zavináče) a adresu hostitelského počítače, popř. dvojtečkou odděleným číslem portu, na kterém probíhá komunikace. Jak uživatelské jméno, tak číslo portu jsou volitelné informace. Adresa hostitelského počítače může být zadána buď doménovým jménem, např. „cs.wikipedia.org“ nebo IP adresou „91.198.174.225“.
- **Cesta zdrojového dokumentu** navazuje na autorizační část a dá se charakterizovat jako posloupnost segmentů vedoucí k cílovému

[7] RDF (ResourceDescription Framework, česky Systém pro popis zdrojů) je používán jako obecná metoda pro modelování informací v různých syntaxích.

dokumentu (koncepčně podobná struktuře adresářů) a mezi sebou oddělená dopředným lomítkem.

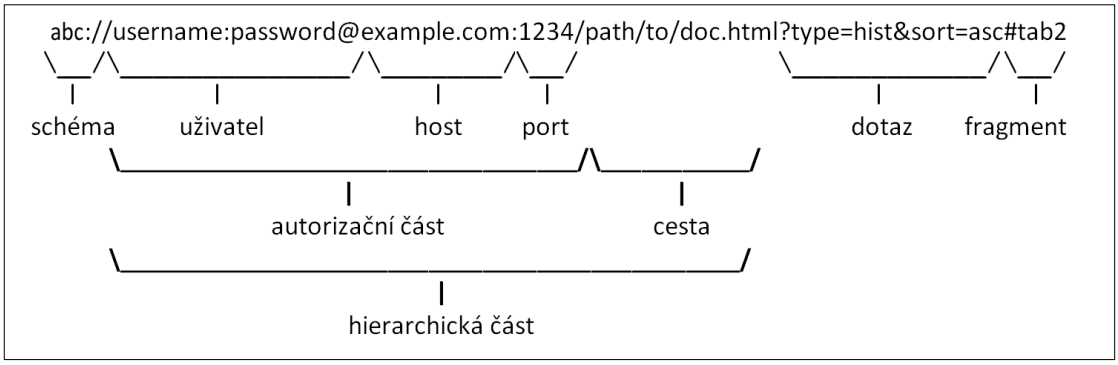
**Dotaz** patří k volitelným částem a slouží ke specifikaci dotazu. Nejčastější použití je při předávání dotazů různým vyhledávacím službám. Jeho syntaxe není přesně specifikována. Obyčejně je však tvořen dvojicemi typu „klíč=hodnota“. Dotaz může vypadat následovně:

```
?key1=value1;key2=value2
```

**Fragment** je stejně jako dotaz nepovinná část. Od zbytku URI je oddělen znakem mřížky „#”. V URI odkazuje většinou na zdroj, který je podřízen jinému primárnímu zdroji.

Obecný tvar URI se tedy dá rozvést do této podoby:

```
<schéma>://<uživatel>:<heslo>@<počítač>:<port>/<cesta>
?<dotaz>#<fragment>
```




**Obrázek 4: URI adresa a její rozdělení na části**

### 1.4.3 Schémata

Jak již bylo řečeno, schéma určuje typ zdroje. V následujícím krátkém přehledu jsou uvedeny pouze typy schémat relevantní k tématu této práce. Všechna uvedená schémata však v dnešní době patří mezi nejvíce užívané typy. Seznam všech trvalých i dočasných schémat se dá nalézt na drese: http://www.iana.org/assignments/uri-schemes.html.

#### 1.4.3.1 Schéma HTTP (Hypertext Transfer Protokol)

Toto schéma je nejpoužívanější a slouží pro výměnu hypertextových dokumentů ve formátu HTML a dalších součástí těchto dokumentů, jako např. obrázků a kaskádových stylů. Jeho syntaxe je:

*http://<počítač>:<port>/<cesta>?<dotaz>#<fragment>*

Počítač v této syntaxi je konkrétní typ daného webového serveru, na kterém se nalézá daný dokument. Port se uvádí pouze v případě, když daný webový server komunikuje na jiném než standartním portu 80. Cesta popisuje průchod složkami k danému dokumentu. Cesta může být i prázdná pokud odkaz ukazuje na hlavní stránku serveru. Dokumenty mají nejčastěji koncovky typu „.htm“ a „.html“.

Dotaz a fragment jsou nepovinné části. Dotaz se na webových stránkách používá k předávání parametrů v případech dynamicky generovaných stránek. Fragment se používá při ukázání na určitou část dokumentu – např. webové stránky http://www.mapy.cz používají fragment pro zobrazení části mapy.

#### 1.4.3.2 Schéma HTTPS (Hypertext Transfer Protokol Secure)

HTTPS je nadstavbou síťového protokolu HTTP, umožňující zabezpečit spojení mezi klientem a serverem před případným odposloucháváním, podvržením dat a zároveň umožňující ověřit identitu protistrany. HTTPS používá pro přenos protokol HTTP, přičemž jsou přenášená data šifrována pomocí protokolu SSL nebo TLS a standardně používá ke komunikaci port 443.

Syntaxe HTTPS je shodná se syntaxí HTTP, kromě typu schématu.

#### 1.4.3.3 Schéma mailto

Od ostatních schémat se liší tím, že neslouží k určení konkrétního informačního zdroje. Mailto schéma je registrován u organizace IANA[8] (Internet Assigned Numbers Authority) a definuje schéma pro protokol SMTP určený pro přenos elektronické pošty (e-mailů).

Syntaxe mailto je nejjednodušší ze všech schémat.

*mailto:e-mailová adresa*

Za dvojtečkou následuje pouze adresa pro elektronickou poštu.

Například: mailto:domorad.vojtech@gmail.com

#### 1.4.3.4 Schéma FTP (File Transfer Protokol)

Syntaxe schématu FTP vychází jako schéma HTTP z obecného tvaru syntaxe URI. Avšak syntaxe schématu FTP neobsahuje část dotazu a fragmentu. Pro přístup k cílovému souboru používá přenosový protokol FTP. Tento protokol slouží pro přenos souborů mezi počítači nebo servery. FTP protokol používá pro komunikaci standardně port 21. Pokud tedy požadujeme jiný port, musíme ho uvést za dvojtečkou po adrese serveru. V případě, že soubory na serveru jsou přístupné komukoliv, nemusíme před adresou serveru uvádět uživatelské jméno ani heslo. V tomto případě se klient přihlásí k serveru jako uživatel *anonymous* a jako heslo použije klientovu emailovou adresu nebo může zůstat prázdné.

[8] IANA je organizace dohlížející na přidělování IP adres, správu kořenových DNS serverů, RFC dokumentů a typů médií pro internetový standart MIME, který umožňuje pomocí elektronické pošty zasílat zprávy s textem obsahující diakritiku a přikládat přílohy v nejrůznějších datových formátech apod.

Syntaxe tohoto schématu je následující:

*ftp://<uživatel>@<počítač>:<port>/<cesta>*

*<cesta>*identifikuje soubor v rámci serveru. Popisuje průchod adresáři k cílovému souboru. Neodkazuje-li soubor, měla by končit lomítkem. Cesta může být prázdná, pokud odkaz vede na kořenový adresář serveru.

Příklady:

ftp://ftp.freebsd.org
ftp://ftp.freebsd.org/pub/FreeBSD/doc/README.txt
ftp://user1:123456@10.0.0.1:121/images/

#### 1.4.3.5 Schéma file

Schéma file je poněkud odlišné od koncepce URI. Slouží totiž k identifikaci souborů v počítači na lokálním disku. V syntaxi se tedy vynechává část specifikující adresu počítače. Má tedy tvar:

*file://cesta*

*Cesta* identifikuje soubor v rámci adresářové struktury na disku v počítači. Popisuje průchod složkami systému až k danému cíli. Vzhledem k tomu, že cesta začíná v kořenovém adresáři, jenž je představován lomítkem, následují za názvem schématu lomítka tři a ne dvě jako u ostatních typů schémat. Odkazuje-li cesta na celou složku, tak by měla končit lomítkem. Cesta může být i prázdná v případě, že ukazuje na kořenovou složku.

Jak bylo řečeno, tak schéma *file* slouží k identifikaci souborů na disku počítače, proto by se nikdy neměly používat na stránkách učených pro Internet.

Příklady:

file:///
file:///c:/users/domorad/
file:///c:/users/domorad/users.html

### 1.4.4 Absolutní a relativní adresy

URL adresa začínající schématem je adresa absolutní. Dané je to tím, že URL adresa zdroje obsahuje kompletní informaci k získání konkrétního dokumentu – například *http://www.literatura.cz/knihy/zdarma/*. Funguje odkudkoliv a ukazuje vždy na stejný cíl (z toho vyplývá název „absolutní“).

Relativní adresa proti tomu obsahuje pouze informaci u umístění zdroje. Určení typu schématu i serveru v této URL adrese chybí – například *kapitola1.html*. Relativní adresa je tedy nějaký zkrácený zápis adresy, který se vždy posuzuje v kontextu stránky, na které se vyskytl. Aby mohla být z relativní adresy vytvořena adresa absolutní, je potřeba zkombinovat danou relativní adresu s adresou stránky, na které se vyskytla. Ve výsledku to znamená, že různé webové stránky se stejnými relativními adresami povedou k různým cílům.

Relativní adresy usnadňují zejména práci autorům webových stránek - nemusejí pokaždé vypisovat kompletní (absolutní) adresy a měnit je kdykoliv se stránky přesunou jinam.

Používání relativních adres se však neomezuje pouze na odkazování v rámci dané úrovně. Relativní adresy umožňují odkazovat v rámci celého webového serveru, tzn. z kořenového adresáře nebo z konkrétního místa (úrovně) výskytu relativního odkazu.

Odkazování v rámci celého webového serveru začíná znakem dopředného lomítka a nahradí celou cestu v URL adrese – schéma i jméno hostitele zůstává. Například pokud na adrese *http://www.literatura.cz/knihy/zdarma/* použijeme relativní odkaz s adresou */clanky/* dostaneme se na adresu *http://www.literatura.cz/clanky/* - vychází se z kořenového adresáře hostitelské domény a mění se pouze cesta.

Odkazování vycházející ze stejné úrovně, na kterém se nachází relativní odkaz, začíná jinými znaky než lomítkem. Skladba odkazu pak funguje odlišně nežli u absolutního odkazování. V prvním kroku se z URL adresy vypustí vše za posledním lomítkem – tzn., vychází se ze složky obsahující daný HTML dokument. V dalším kroku se na konec URL odkazu přidá cesta uvedená v odkazu. Pokud odkaz začíná dvěma tečkami, pak odkazuje na složku, která se nachází o úroveň výše.

Následující tabulka ukazuje přehled používaných typů relativních odkazů. Pro názornost řekněme, že tyto odkazy jsou uvedeny např. v HTML dokumentu na webové stránce s adresou *http://www.literatura.cz/knihy/zdarma/index.html*. URL odkazy, jež by se na dané

stránce mohly vyskytnout, jsou uvedeny vlevo, vpravo jsou pak uvedeny URL adresy, na které dokazují.

**Tabulka 4: Ukázky skládání URL**

| **Relativní URL** | **Výsledné (absolutní) URL** |
|---|---|
| *obsah.html* | *http://www.literatura.cz/knihy/zdarma/obsah.html* |
| *prog/java.html* | *http://www.literatura.cz/knihy/zdarma/prog/java.html* |
| */* | *http://www.literatura.cz/* |
| */clanky/prehled.html* | *http://www.literatura.cz/clanky/prehled.html* |
| *../placene/top10.html* | *http://www.literatura.cz/knihy/placene/top10.html* |
| *#sekce2* | *http://www.literatura.cz/knihy/zdarma/index.html#sekce2* |

# 2 Rešerše současných nástrojů

Údržba webu a všech URL odkazů, které se na daném webu nacházejí, není vždy jednoduchá věc. Čím více dokumentů webový server obsahuje, tím větší je pravděpodobnost chyby, že se při zápisu některého odkazu vývojář zmýlí. Další chybou, která může nastat, je že stránku, na kterou adresa odkazuje, provozovatel jiného serveru odstraní. Toto nejenomže snižuje profesionalitu webových stránek, ale zároveň i obtěžuje návštěvníky, kteří navštíví tyto stránky. Proto je vhodné jednou za čas použít nějaký nástroj, umožňující kontrolovat funkčnost odkazů na webových stránkách.

V dnešní době existuje na trhu několik aplikací určených ke kontrole těchto odkazů. Tyto aplikace lze rozdělit na dva typy. Jednou skupinou jsou desktopové aplikace, druhou jsou online nástroje.

## 2.1 Online nástroje

Všechny online nástroje mají jednu velkou výhodu. Nemusejí se nikam instalovat a jsou dostupné odkudkoliv. Bohužel jim musím vytknout jejich pomalé zpracování a jednoduchý nepříliš přehledný výpis výsledků. Pomalé zpracování je způsobeno jednovláknovým zpracováváním nalezených odkazů. Vícevláknové zpracování je samozřejmě možné, ale z důvodu šetření serverových prostředků není aplikováno. V rámci celého internetu je těchto nástrojů nepřeberné množství. Avšak kromě několika funkčních výjimek jako například W3C Link Checker nestojí většina z nich ani za zmínku.

Existuje však i několik placených verzí online nástrojů (služeb), které kontrolují webové stránky průběžně a posílají statistiky výsledků a varování v přehledných tabulkách a grafech na uvedený mail. Mezi takové patří např. LinkTiger od stejnojmenné společnosti. Z důvodu registrace uživatele a vyplnění údajů o platební kartě před aktivací 15 denní zkušební verze nebyl tento nástroj otestován.

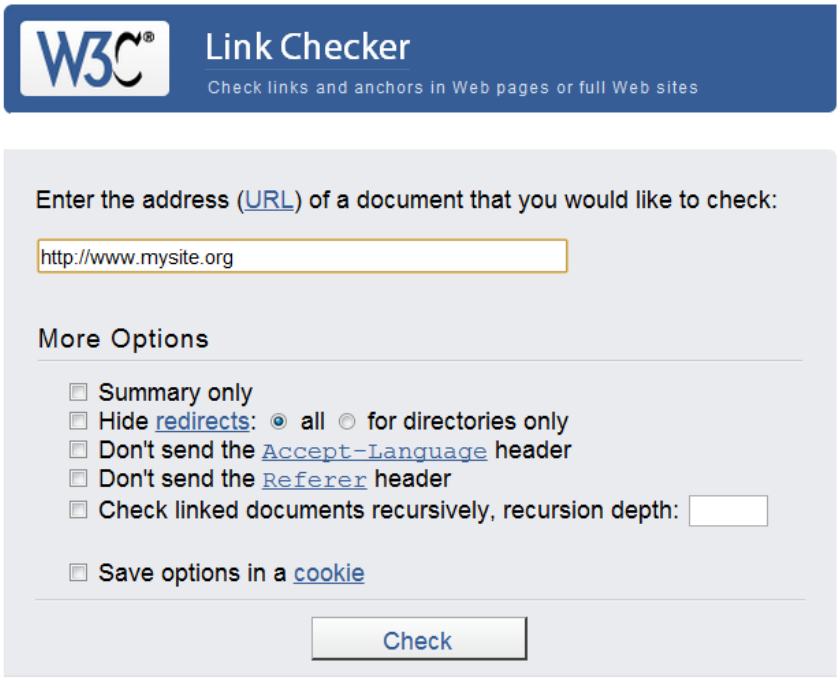
### 2.1.1 W3C Link Checker

Tento online nástroj pro kontrolu odkazů patří mezi nejstarší. Jeho první verze byla publikována již v roce 1998. Jak už z názvu napovídá, tento validátor spadá přímo pod W3C

konsorcium. Je možné kontrolovat odkazy v HTML, XHTML dokumentech nebo soubor kaskádových stylů. Kontroluje, zdali nejsou relativní odkazy v rámci daného dokumentu definovány několikrát. Varuje v případě přesměrování a umožňuje kontrolovat odkazy rekurzivně. Je napsán v programovacím jazyku Perl pod licencí W3C Software Licence. Zdrojové kódy jsou veřejně dostupné.

**Tabulka 5: Shrnutí vlastností programu W3C Link Checker**

| **W3C Link Checker** | **http://validator.w3.org/checklink** |
|---|---|
| Verze | 4.81 |
| Datum vydání | 16. 10. 2011 |
| Podporované jazyky | Angličtina |
| Operační systémy | nezávislé na platformě – pouze HTML interface |
| Autor programu | W3C |
| Programovací jazyk | Perl |
| Velikost | Není známo -online nástroj |
| Licence | Open Source |
| Typy kontrolovaných odkazů | HTTP, HTTPS, FTP |
| Výhody | základní možnosti nastavení<br>kontrola duplicitně vytvořených relativních kotev v rámci jednoho dokumentu<br>varování při přesměrování<br>zdrojové kódy jsou veřejně dostupné |
| Nevýhody | pomalý<br>nastavení pouze základních možností |




**Obrázek 5: W3C Link Checker**

## 2.2 Desktopové aplikace

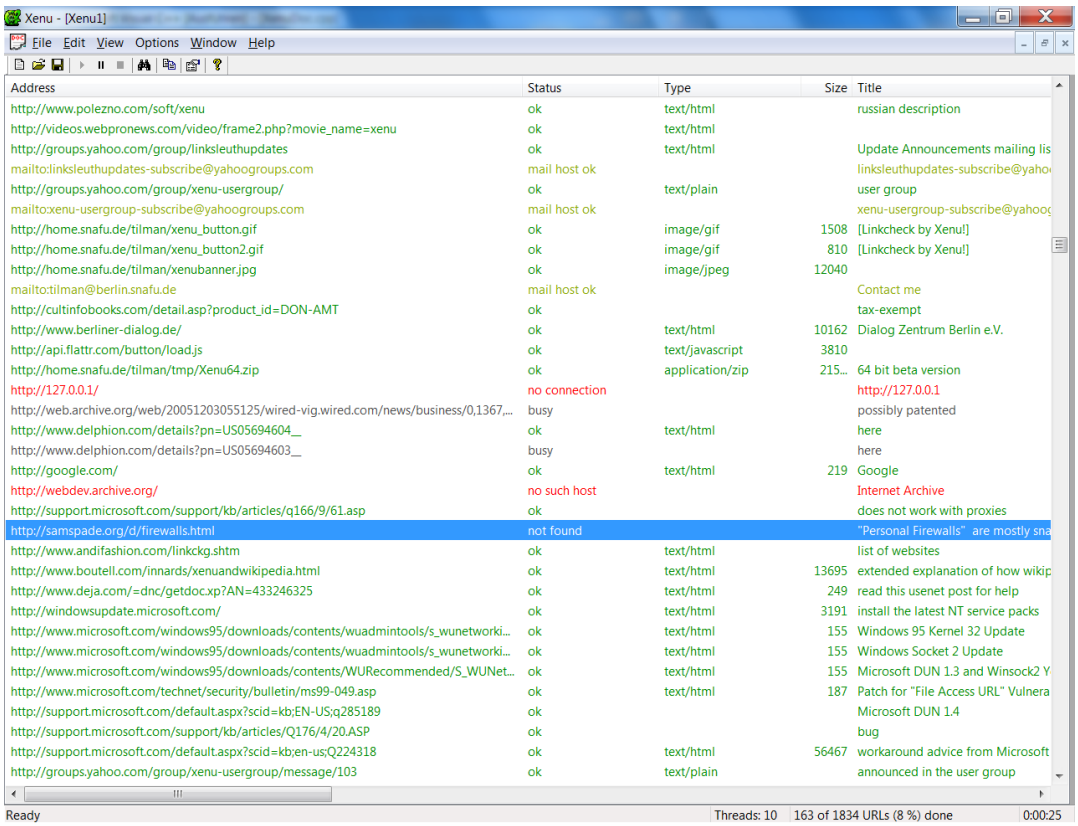
Na rozdíl od online řešení poskytují mnohem větší pohodlí při zobrazování a zpracování výsledků. Mají velké množství různých nastavení a práce s nimi jde rychle a velice snadno. První dva popsané nástroje patří mezi volně šiřitelné a tudíž i bezplatné. Zbylé dva nástroje patří do komerční sféry a jejich bezplatné užívání je omezeno pouze na 30 dní.

### 2.2.1 Xenu's Link Sleuth

Tento jednoduchý nástroj, patří mezi nejlepší bezplatné varianty. Po instalaci a spuštění aplikace se uživateli zobrazí přehledné okno, ve kterém po spuštění zobrazí okno s tipy a triky k užívání programu. Před spuštěním kontroly si může uživatel nastavit filtry a maximální počet souběžných vláken. Při kontrole program v případě nutnosti autorizace uživatele požádá o zadání uživatelského jména a hesla nebo případně i certifikátu. Jsou podporována schémata typu HTTP, HTTPS, gopher a file. V případě potřeby přerušení kontroly odkazů, je možné rozpracovaný projekt uložit a pokračovat od posledního zkontrolovaného odkazu. Spolu s rozpracovaným projektem je uloženo i veškeré nastavení. Jednou z mála věcí, která se mu však musí vytknout, je nemožnost shrnutí výsledů do stromové struktury - zkontrolované odkazy nelze rozlišit podle cesty, na které byly nalezeny.

**Tabulka 6: Shrnutí vlastností programu Xenu's Link Sleuth**

| **Xenu's Link Sleuth** | **http://home.snafu.de/tilman/xenulink.html** |
|---|---|
| Verze | 1.3.8 |
| Datum vydání | 4. 8. 2010 |
| Podporované jazyky | Angličtina |
| Operační systémy | MS Windows od verze 98 výše |
| Autor programu | Tilman Hausherr |
| Programovací jazyk | C++ |
| Velikost | 453kB |
| Licence | Freeware |
| Typy kontrolovaných odkazů | HTTP, HTTPS, FTP, Gopher, file |
| Výhody | jednoduchý a přehledný uživatelský interface<br>možnost uložení rozpracovaného projektu<br>podpora autentizace uživatele a SSL |
| Nevýhody | nelze zobrazit nefunkční odkazy podle adresy nalezeného dokumentu |




**Obrázek 6: Xenu's Link Sleuth**

### 2.2.2 LinkChecker

Dalším velice zdařeným softwarem, který je též zdarma, je LinkChecker. Tento software je stále vyvíjen a zdokonalován už od roku 2000. Je rozšířen na platformy Windows, Linux (Debian) a Mac OS (X). Pro první dvě zmíněné platformy je k dispozici v aktuální verzi 7.9, pro platformu Mac OS pouze ve verzi 7.0. Stejně jako Xenu's Link Sleught umí požádat o autorizaci uživatele v případě zabezpečených odkazů pro schémata HTTP, FTP nebo Telnetu. Pro práci s programem jsou uživateli k dispozici 3 režimy. První možnost je skrze příkazový řádek, další pak je práce skrze klasické grafické uživatelské prostředí a třetí možností je tzv. Common Gateway Interface (CGI). Ač se může zdát, že tato aplikace je téměř dokonalá, není tomu tak. Mezi její hlavní nedostatky patří zejména nepropracované grafické uživatelské prostředí. V něm není možná jakákoliv filtrace nalezených odkazů, či nastavení počtu souběžně spuštěných vláken, zpracovávající kontrolované odkazy. Všechna tato nastavení jsou však dostupná při použití aplikace z příkazového řádku.

**Tabulka 7: Shrnutí vlastností programu LinkChecker**

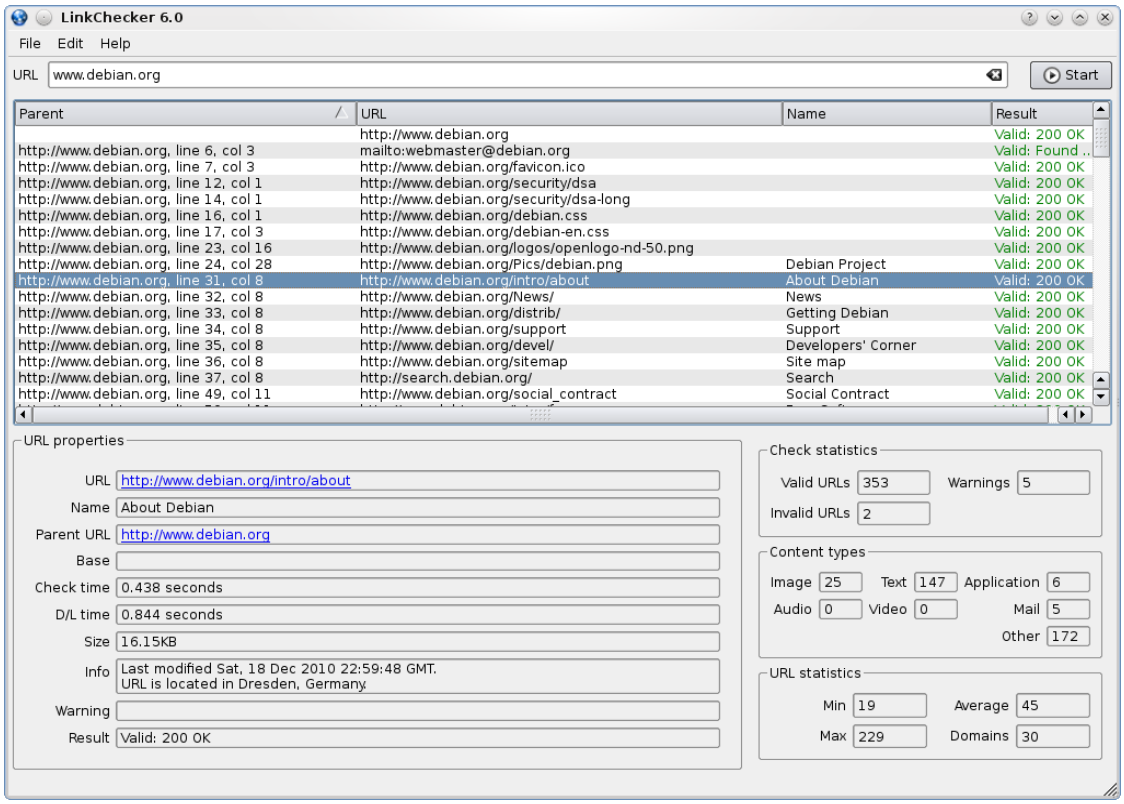
| **LinkChecker** | **http://linkchecker.sourceforge.net/** |
|---|---|
| Verze | 7.9 |
| Datum vydání | 10. 6. 2012 |
| Podporované jazyky | Angličtina |
| Operační systémy | MS Windows od verze 98, Debian Linux, Mac OS X (pouze verze 7.0) |
| Autor programu | „calvin“ |
| Programovací jazyk | C, Python, Yacc |
| Velikost | 11 470kB |
| Licence | GNU General Public License version 2.0 (GPLv2) |
| Typy kontrolovaných odkazů | HTTP, HTTPS, FTP, Telnet, mailto, news, nntp, file |
| Výhody | kontinuální vývoj aplikace<br>3 typy uživatelského prostředí – CLI, GUI, CGI<br>rozšíření na Windows, Linux a Mac OS<br>podpora autentizace uživatele, SSL |
| Nevýhody | nepropracované grafické uživatelské rozhraní, nalezené odkazy lze přidat do seznamu adres ignorovaných pouze ručním zapsáním |

```
URL         'index.ko.html'
Name        'Korean'
Parent URL  http://www.debian.de/, line 248, col 1
Real URL    http://www.debian.de/index.ko.html
Check Time  3.105 seconds
Info        Last modified Fri, 27 Aug 2004 14:13:00 GMT
Result      Valid: 200 OK

URL         'index.hr.html'
Name        'Croatian'
Parent URL  http://www.debian.de/, line 249, col 1
Real URL    http://www.debian.de/index.hr.html
Check Time  3.109 seconds
Info        Last modified Fri, 27 Aug 2004 13:49:47 GMT
Result      Valid: 200 OK

Thats it. 0 errors in 134 links found
```

**Obrázek 7: LinkChecker – rozhraní příkazového řádku**




**Obrázek 8: LinkChecker – grafické uživatelské rozhraní**

## 2.2.3 Web Link Validator

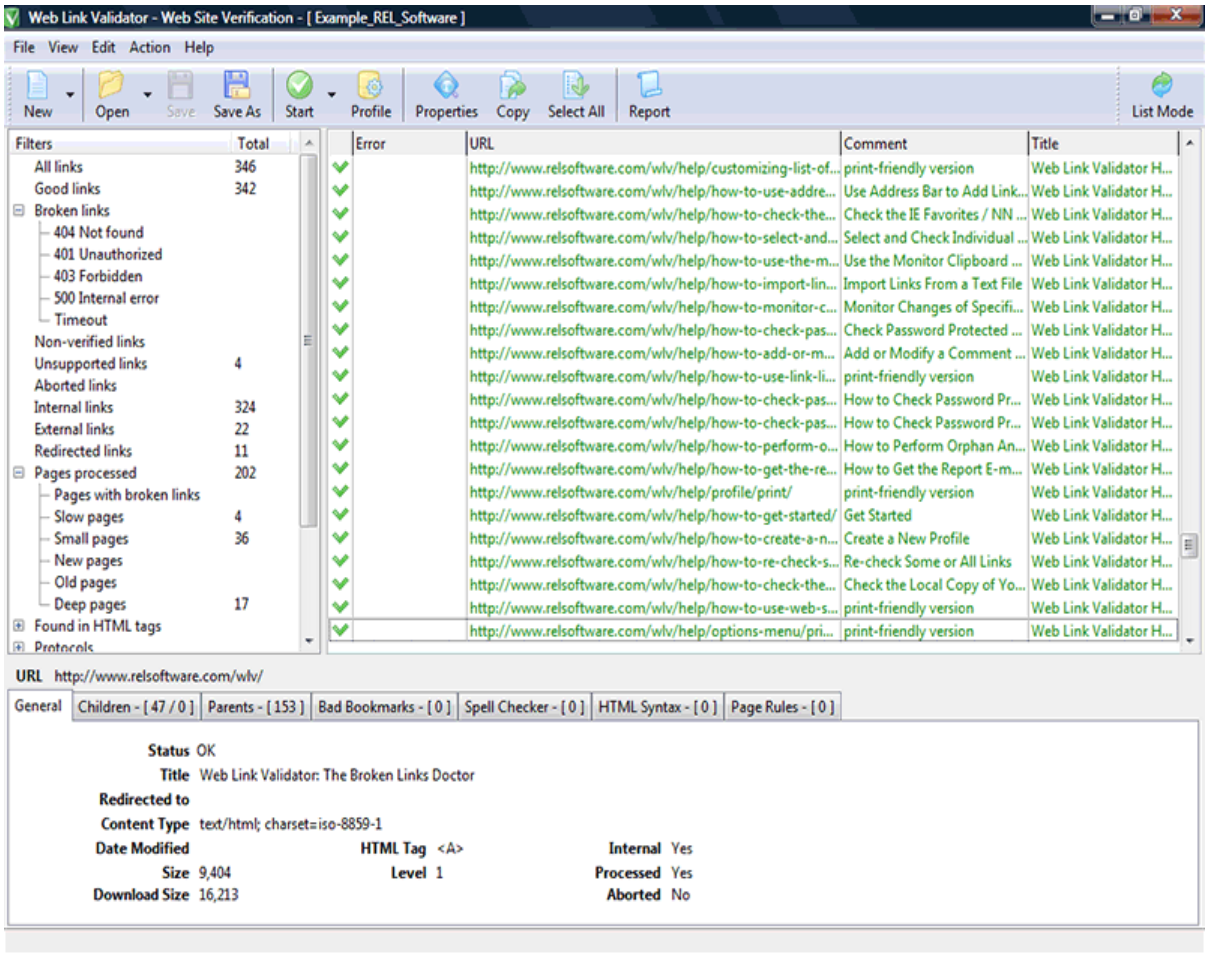
Doposud představený software měl minimálně jednu věc společnou - cenu softwaru, která byla zdarma. Web Link Validator od společnosti REL Software patří do skupiny komerčního softwaru. Jeho cena začíná na hranici 145 amerických dolarů pro jednoho uživatele a je omezena možným počtem kontrolovaných odkazů, v tomto případě je to

maximálně 3000 odkazů kontrolovaných najednou. Pokud by chtěl uživatel neomezený počet kontrolovaných linků, tak zaplatí téměř 1200 amerických dolarů.

Po instalaci a spuštění softwaru jsou vidět 3 panely. Jeden navigační, který slouží k rozdělení nalezených odkazů do definovatelných kategorií. Defaultně je přednastaveno několik typů kategorií, jako jsou např. všechny odkazy, v pořádku, špatné, atd. Dále lze odkazy dělit do kategorií podle návratového kódu, typu schématu nebo kontrolovaného obsahu. Pro spuštění kontroly odkazů jsou k dispozici 2 režimy. Rychlý, kde stačí zadat pouze adresu webu. A druhý, založení nového projektu, kde má uživatel možnost nastavení všeho na co si jen vzpomene – od autentizace uživatele po zaslání emailu při dokončení kontroly. Co se rychlosti týče, program testuje odkazy o 30 procent rychleji nežli konkurenční software. Autor práce také zjistil, že software při transformování odkazů v několika případech nefungoval správně. Tento problém se vyskytl při nalezení odkazu ukrytého uvnitř javascriptové funkce, který byl ve zdrojovém kódu stránky ohraničen znakovou entitou dvojitých uvozovek. Program pak tento odkaz převedl jako *absolutní_adresa_hosta/"odkaz"*.

**Tabulka 8: Shrnutí vlastností programu Web Link Validator**

| **Web Link Validator** | http://www.relsoftware.com/ |
|---|---|
| Verze | 5.5 (build 553) |
| Datum vydání | 27. 10. 2010 |
| Podporované jazyky | Angličtina, Němčina, Španělština, Italština, Francouzština, Švédština, Holandština |
| Operační systémy | MS Windows XP/Vista/7/Server 2003/2008 |
| Autor programu | REL Software |
| Programovací jazyk | neznámý |
| Velikost | 3 652kB |
| Licence | Shareware |
| Typy kontrolovaných odkazů | HTTP, HTTPS, FTP, file, mailto |
| Výhody | jednoduchost<br>technická podpora, záruka ze strany dodavatele softwaru<br>průběžné monitorování definovaných odkazů<br>export výsledků |
| Nevýhody | cena<br>špatná transformace odkazů |




**Obrázek 9: Web Link Validator**

## 2.2.4 HTML Link Validator

HTML Link Validator se díky dlouholetému vývoji řadí mezi špičku v tomto oboru. Za posledních několik let se však jeho vývoj téměř zastavil. Sice byla v minulém roce vydána nová verze, ale ta jen přidává možnost nastavení zpoždění před kontrolou odkazu. Právě z tohoto důvodu mu lze vytknout jeho rozšíření pouze pro operační systémy Windows XP a jeho předchůdce. V novějším operačním sytému sice spustit lze, nicméně při kontrole samotných odkazů aplikace „zamrzne“. Tento software se řadí do skupiny komerčních a jeho cena je 35 dolarů pro jednoho uživatele. Program si však lze před koupí vyzkoušet na 30 dní zdarma.

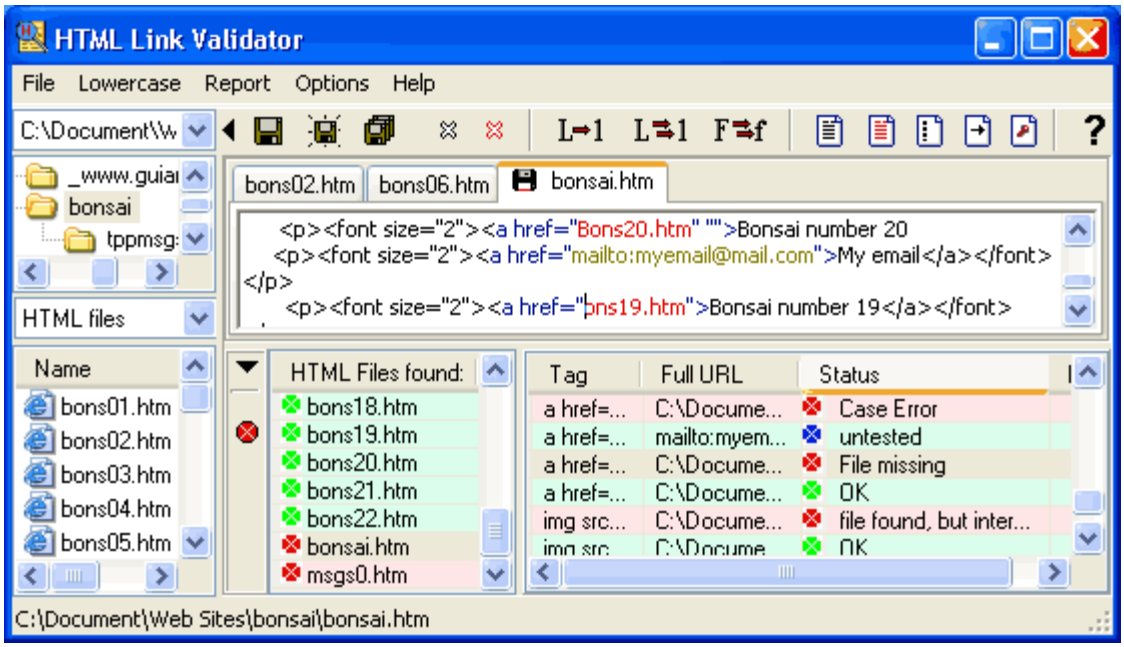
Při prvním spuštění programu se může na první pohled zdát, že grafická stránka programu je poněkud chaotická. Nicméně všechny nejdůležitější funkce jsou umístěny právě na hlavním panelu. Na levé straně je okno se zobrazením kořenové struktury místního

primárního pevného disku, pod hlavní nástrojovou lištou jsou základní předvolby – kontrola projektu uloženého v počítači nebo kontrola HTML dokumentů na webovém serveru. Ve spodní části okna se nachází dvě menší okna, sloužící k výpisu odkazů a dokumentů, ve kterých byly tyto odkazy nalezeny.

Podrobnější nastavení možností je přístupné z hlavního panelu. Zde je k dispozici například i nastavení parametrů procesů kontrolujících odkazy, list všech kontrolovaných a nekontrolovaných odkazů. Mezi výhody této aplikace patří vestavěný prohlížeč zdrojového kódu. Program sám o sobě všechny nalezené dokumenty ukládá na lokální disk v počítači, takže i po ukončení kontroly a odpojení od internetu je možné zdrojové kódy webových stránek procházet.

**Tabulka 9: HTML Link Validator**

| **HTML Link Validator** | http://lithopssoft.com/hlv/ |
|---|---|
| Verze | 4.52 |
| Datum vydání | 17. 2. 2011 |
| Podporované jazyky | Angličtina |
| Operační systémy | MS Windows 95/98/Me/2000/XP |
| Autor programu | Lilhops Software |
| Programovací jazyk | neznámý |
| Velikost | 1 000kB |
| Licence | Shareware |
| Typy kontrolovaných odkazů | HTTP, HTTPS, FTP, mailto |
| Výhody | rozhraní GUI a CLI<br>přehledná filtrace nalezených odkazů<br>rychlost kontroly |
| Nevýhody | podpora pouze starších OS od společnosti Microsoft<br>nepřehledné grafické uživatelské prostředí |




**Obrázek 10: HTML Link Validator**

# 3 Analýza a návrh aplikace

V této kapitole je zanalyzován zadaný úkol a popsán obecný návrh aplikace.

## 3.1 Proč se zabývat tvorbou dalšího softwaru

Dříve než se pustíme do analýzy zadaného úkolu, je potřeba si položit následující otázku: „Proč vůbec začínat s vývojem nového softwaru na kontrolu odkazů, když už existuje několik alternativ a všechny plní základní funkci kontroly odkazů?

V první chvíli, se může zdát, že odpověď na tuto otázku je složitá. Nicméně, je tomu přesně naopak. Současný software, je schopen rychlého získání a kontroly webových odkazů, avšak přehledné zpracování výsledků je slabší stránkou většiny nástrojů. Proto je mezi hlavní cíle této diplomové práce kladen důraz právě na přehlednou prezentaci a práci s chybnými odkazy.

V několika málo programech, které byly popsány v předchozí kapitole, je už na první pohled zřejmé, co v nějaké té konkrétní aplikaci chybí. Jeden z těchto nástrojů, dokonce ani neumožňoval filtrovat výsledky skrze grafické uživatelské rozhraní, což při testování portálu obsahujícím tisíce, či až desítky tisíc odkazů odradí uživatele od používání aplikace a sáhnutí po jiné alternativě.

## 3.2 Analýza zadaného úkolu

Před samotným návrhem aplikace je potřeba vytvořit představu o funkčnosti programu a na jejím základě vytvořit konkrétní řešení zadaného úkolu.

Na první pohled je jasné, že vyvíjená aplikace bude muset interagovat s uživatelem a uvnitř aplikace bude probíhat výměna dat s webovým serverem. Velkou roli v samotném řešení pak bude hrát zpracování odkazů nalezených na webových stránkách.


**Obrázek 11: Schéma návrhu**

Výsledný vývojový proces aplikace se bude skládat z několika základních stavebních bloků, které jsou uvedeny v následujícím grafu.


**Obrázek 12: Vývojový proces navrhované aplikace**

Interakce s uživatelem bude probíhat v krocích nastavení předvoleb, spuštění kontroly a filtrování výsledků. Před spuštěním samotné kontroly musí uživatel nastavit několik povinných vstupních údajů:

- Adresu kontrolovaného serveru
- Úroveň do jaké hloubky bude kontrola prováděna

Volitelně bude mít uživatel k dispozici další možnosti nastavení:

- Počet souběžně běžících procesů, kontrolujících odkazy
- Nastavení času mezi kontrolami jednotlivých odkazů
- Odkazy, které chce/nechce uživatel kontrolovat
- Kontrola odkazů nalézajících se na jiném serveru nežli kontrolovaném

Bloky získání, kontroly a uložení odkazů lze označit jako jádro aplikace, o jejichž práci bude mít uživatel pouze informativní přehled v podobě počtu zkontrolovaných odkazů, či jejich výpisu do tabulky.

Získání odkazů bude fungovat na základě stažení zdrojového kódu dotyčného dokumentu a pomocí regulárních výrazů procházet webové stránky a vyhledávat všechny odkazy na webových stránkách.

Při kontrole odkazů bude na každý jednotlivý odkaz zaslán požadavek a ze zprávy s odpovědí bude vyňat stavový kód a zpráva náležející tomuto kódu.

# 4 Vývoj aplikace

## 4.1 Programovací jazyk a vývojové prostředí

Kvůli nezávislosti na platformě a předchozí znalosti jazyka, zvolil autor práce objektově orientovaný programovací jazyk Java, který je jedním z nejpoužívanějších programovacích jazyků na světě.

Tento jazyk je oceňován především začínajícími programátory pro svou jednoduchou syntaxi oproti syntaxi jazyka C a C++. Velkou výhodou Javy je její *hardwarová nezávislost* - je překládaná do speciálního mezikódu (tzv. *bytecode*), který je na konkrétním počítači nebo zařízení (PC, mobilní telefon apod.) interpretován, příp. za běhu překládán do nativního kódu (tzv. *JIT - Just-In-Time* compilerem). Programátor tedy může napsat program v jazyce Java například na PC s operačním systémem Windows a spustit jej na PC s Linuxem. Jediným omezením je nutnost nainstalovaného prostředí pro běh aplikací (JRE – Java Runtime Environment), které poskytuje základní knihovny a další komponenty potřebné pro spuštění programu.

Základní vlastnosti Javy jsou:

- Hardwarová nezávislost a přenositelnost
- Jednoduchost syntaxe
- Objektová orientace
- Robustnost
- Interpretace
- Vícevláknové zpracování

Oproti ostatním jazykům, které provádějí tzv. statickou kompilaci (jako např. C++) je Javě vytýkán pomalý náběh aplikace. To je právě působeno kompilací programu před každým spuštěním. Další hlavní nevýhodou je větší paměťová náročnost a to zejména u jednodušších programů.

Při vývoji aplikace bylo použito vývojové prostředí Eclipse a NetBeans IDE. Eclipse je bezplatná open source platforma, primárně určená pro programování v jazyce Java. Oproti ostatním vývojovým prostředím v Javě, jako například Netbeans, je filozofie Eclipse úzce svázána právě s rozšiřitelností pomocí pluginů. V základní verzi obsahuje Eclipse pouze

integrované prostředky pro vývoj standardní Javy jako kompilátor a debugger, ale neobsahuje například nástroj pro vizuální návrh grafických uživatelských rozhraní desktopových aplikací. Tato platforma byla použita při vývoji jádra aplikace, kde se dbalo na funkčnost aplikace - vyhledávání, testování a ukládání odkazů.

V druhé fázi vývoje aplikace, kdy již jádro aplikace fungovalo, bylo potřeba vytvořit grafické uživatelské prostředí. Grafická úprava aplikace byla vyvíjena v prostředí Netbeans IDE. Oproti vývojovému prostředí Eclipse se právě liší implementovaným návrhovým prostředím pro grafické uživatelské rozhraní. Rysy jsou ale velice podobné. Jedná se také o open source projekt jež byl vyvinut primárně pro vývoj aplikací v jazyce Java, a který lze pomocí modulů rozšířit.

Důležité je podotknout, že aplikace byla vyvíjena pod nejnovější verzí Java Standard Edition 7. Toto programovací rozhraní na rozdíl od předchozích verzí neopravuje jen chyby, ale přináší i spoustu nových funkcí, jako je např. možnost využití příkazu *switch* pro použití s textovými řetězci.

## 4.2 Strukturování projektu

Z hlediska logiky a existence předpokladu, že bude projekt v budoucnosti rozšiřován, bylo vhodné projekt rozumně strukturovat. Základní struktura projektu je znázorněna na Obrázek 13: Základní struktura projektu.

V projektu bylo vytvořeno 5 balíčků:

- *data_objects*

  Zde jsou uloženy nadefinované třídy pro práci s datovými objekty, jako například úložiště s odkazy, konstantní hodnoty, filtry a předvolby.
- *frames*

  Balíček s nadefinovaným grafickým uživatelským rozhraním, ve kterém se nacházejí jednotlivá zobrazovaná okna.
- *helpers*

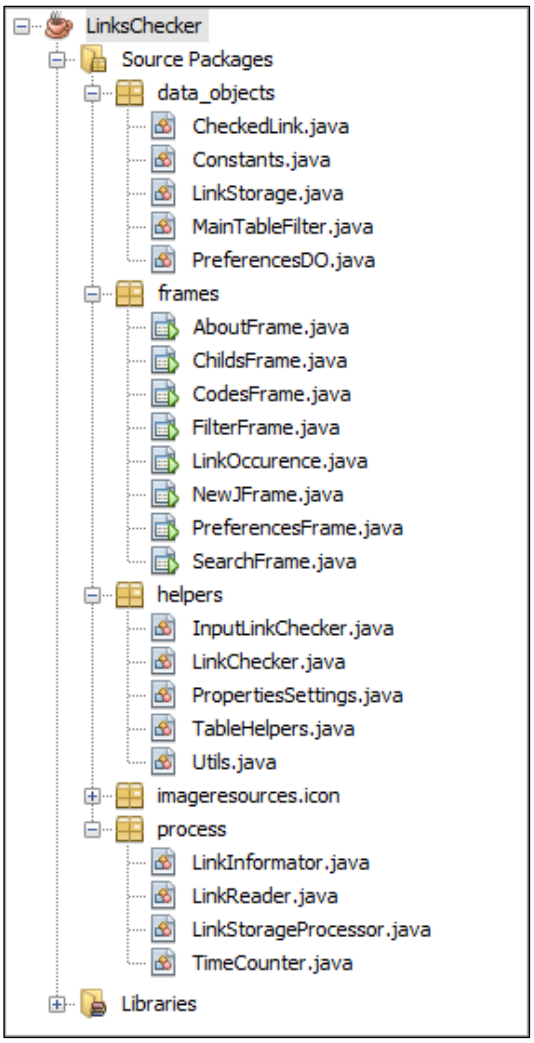
  Tento balíček obsahuje všechny pomocné třídy s nadefinovanými metodami používanými při kontrole odkazů. Mezi tyto „pomocníky“ patří například různé transformace odkazů před zpracováním či kontrola odkazů.

- *imageresources.icon*

  V tomto balíčku jsou uloženy pouze ikony jednotlivých tlačítek používaných v programu.

- *process*

  Jeden z hlavních balíčků, ve kterém se provádí sběr odkazů na webových stránkách, ukládání odkazů či různé informativní operace zobrazující uživateli průběh kontroly.




**Obrázek 13: Základní struktura projektu**

# 5 Jádro aplikace

Jádrem samotné aplikace se dají označit balíčky data_objects, helpers a process. V nich se shromažďují, kontrolují a ukládají jednotlivé nalezené odkazy.

## 5.1 Zpracování odkazů

Počátek zpracování odkazů začíná ve třídě *LinkStorageProcessor*, jejíž kód je uveden v příloze A. Z URL adresy zadané uživatelem se vytvoří objekt definovaný třídou *LinkReader*, popsanou v kapitole 5.2, jehož metoda *run()* vyhledává jednotlivé odkazy ve zdrojovém kódu dokumentu.

Z nalezených odkazů se nejdříve vytvářejí objekty definované třídou *CheckedLink* a takto vytvořené objekty se ukládají do datové struktury typu HashMap v objektu *LinkStorage*. Tyto objekty jsou pojmenované jako *uncheckedLinks*, jejímž klíčem je absolutní URL adresa odkazu nalezeného v atributu tagu.

Dále se v třídě *LinkStorageProcessor* vytváří uživatelem nadefinovaný počet objektů typu *LinkReader*, což jsou ve výsledku simultánně běžící vlákna. Těchto objektů je vytvořeno přesně tolik, kolik jich uživatel nadefinoval v předvolbách aplikace před samotným spuštěním kontroly. Po vykonání činnosti vlákna *LinkReader*, tzn. po kontrole odkazu a jeho přesunutí z hash mapy *uncheckedLinks* do hash mapy *checkedLinks* objekt (vlákno) zaniká a uvolňuje tím místo pro vytvoření a zpracování dalšího nalezeného odkazu z hash mapy *uncheckedLinks*. Tento proces se provádí do té doby dokud hash mapa *uncheckedLinks* není prázdná. Každé toto vlákno, vykonává metodu *run(), ve* které se program snaží spojit s webovým serverem a požádat ho o zdrojový kód dokumentu v dotyčném odkazu a prohledat jeho obsah pro výskyt nových odkazů.

## 5.2 Třída LinkReader

Tato třída provádí procházení obsahu HTML dokumentu dané URL adresy a vyhledává odkazy uložené v atributech tagů.

Samotný proces kontroly se provádí vytvořením objektu třídy typu URL z textového řetězce, který reprezentuje absolutní URL adresu.

```
URL url = new URL(currentLink.getAbsoluteUrl());
```

Po vytvoření objektu *url* se zavolá metoda *checkLink*() z třídy *LinkChecker* z balíčku *helpers*. Tato metoda provede kontrolu, zda kontrolovaná adresa používá schéma HTTP nebo HTTPS a pokusí se získat stavový kód odpovědi a k němu náležející stavové hlášení.

```
LinkChecker.checkLink(parentLink);
```

```
public static void checkLink(CheckedLink link) throws IOException{
   URL url = new URL(link.getTransformedUrl());
   URLConnection urlC = url.openConnection();

   if(url.getProtocol().equals("http") ||
      url.getProtocol().equals("https")){

      HttpURLConnection httpUrlC = (HttpURLConnection) urlC;
      httpUrlC.setInstanceFollowRedirects(false);
      link.setCode(httpUrlC.getResponseCode());
      link.setMessage(httpUrlC.getResponseMessage());

      if (Constants.LINKS_REDIRECTION_CODES_SET.contains(
      Integer.toString(link.getCode()))){
            link.setRedirect(httpUrlC.getHeaderField("Location"));
      }
   }
}
```

Z objektu *url* je volána metoda *openStream()*, která zkouší získat obsah webového dokumentu z absolutní adresy URL. Obsah je procházen po řádcích a na každý řádek je aplikován regulární výraz, který nalezne linky odpovídající tomuto předpisu. Při procházení obsahu řádku se zaznamenává číslo řádku a sloupec, kde byl nalezen tag atributu.

```
content = new BufferedReader(new InputStreamReader(url.openStream()));

while(matcher.find()){
   int column = matcher.start();
   String tag = matcher.group(1);
   String tagAttribute = matcher.group(2);
   String foundLink = matcher.group(3);
   foundLink = foundLink.replace("\"", "");
   foundLink = foundLink.replace("\'", "");
   processFoundLink(foundLink,column,lineNumber,tag,tagAttribute);
}
```

Nalezený odpovídající řetězec se skládá ze tří skupin, kde první skupina odpovídá jménu atributu, druhá atributu tagu a třetí pak odpovídá nalezenému odkazu. Protože jsou hodnoty atributů odkazu na webových stránkách vkládány do uvozovek, je nalezený řetězec URL adresy ohraničen také uvozovkami. Kvůli lepší práci s adresou jsou uvozovky odstraněny. Pro každou nalezenou URL adresu je v metodě *processFoundLink()* vytvářen objekt typu *CheckedLink*. Z adresy aktuální stránky a nalezeného odkazu se pomocí metody *checkLinkUrlAndTranform()* ve třídě *LinkChecker* převede odkaz na absolutní URL adresu a společně s informacemi *message, column, link, tag, tag atribute* je uložena do vytvořeného objektu typu *CheckedLink*. Tento objekt se poté přidá mezi potomky právě prohledávané URL adresy. Před vytvořením objektu se však ještě kontroluje, zdali není vytvořen objekt se stejným primárním klíčem (absolutní adresou odkazu). Tímto způsobem je zaručeno uložení unikátních adres.

```
String absoluteUrl = LinkChecker.checkLinkUrlAndTranform(currentLink,
        foundLink);
if(!LinkStorage.isInCheckedLinks(transformedUrl) &&
        !LinkStorage.isInUncheckedLinks(transformedUrl)){

   CheckedLink link = new CheckedLink(foundLink, parentLink);
   link.setId(LinkStorage.getNextId());
   link.setMessage(Constants.TEXT_NOT_CHECKED);
   link.setColumn(column);
   link.setLine(lineNumber);
   link.setTag(tag);
   link.setTagAttribute(tagAttribute);
   link.setTransformedUrl(transformedUrl);
   parentLink.addChild(link);
        ...
```

Kontrola odkazů probíhá pouze u odkazů se schématem HTTP a HTTPS. Odkazy s ostatními typy schémat jsou uloženy do objektu třídy *LinkStorage* mezi zkontrolované odkazy (*checkedLinks*) a je k nim přiřazena zpráva o nepodporovaném typu schématu. Odkazy se schématem HTTP a HTTPS jsou přidány do objektu třídy *LinkStorage*, mezi nezkontrolované odkazy (*uncheckedLinks*).

Tímto končí zpracování ve třídě *LinkReader* a dále pokračuje zpracovávání nezkontrolovaných odkazů (*uncheckedLinks*) ze třídy *LinkStorageProcessor*.

### 5.2.1 Získávání odkazů

Získávání odkazů z webových stránek, je řešeno skrze regulární výrazy. Při vytváření regulárního výrazu bylo vycházeno ze všech HTML tagů, jejichž atribut obsahuje odkaz. Seznam těchto tagů je uveden v následující tabulce:

**Tabulka 10: Seznam vyhledávaných tagů**

| Tag | Atribut | Použití |
|---|---|---|
| a | href | hypertextový odkaz |
| base | href | základní umístění, zacílení |
| form | action | formulář |
| frame | src | rám, vymezení prostoru pro načtení jiné stránky |
| iframe | src | plovoucí rám |
| img | | |
| link | href | spojitost s jiným souborem |
| script | src | zápis skriptu |
| source | src | určení více zdrojů médií v HTML5 |

Na základě všech těchto tagů a jejich atributů byl sestaven následující regulární výraz, který umožňuje zaznamenat URL odkaz v analyzované stránce včetně HTML tagu a atributu tohoto tagu.

```
<(a|base|form|frame|iframe|img|link|script|source)\b[^<>]*?
\b\s*(action|href|src)\s*=\s*((?:"[^"]*"|'[^']*'))
```

Z reverzní analýzy tohoto regulárního výrazu je možné zjistit, že řetězec znaků, se kterým se shoduje, musí:

- začínat znakem "<"
- následuje jeden z uvedených HTML tagů:
  *a, base, form, frame, iframe, img, link, script, source*
- nenásleduje znak "<" nebo ">"
- následuje libovolný počet bílých znaků (mezera)

- dále musí souhlasit jeden z požadovaného typu atributu tagů:

  *action, href, src*
- další podmínkou je znak "=", obklopený libovolným počtem mezer
- následuje přečtení odkazu, uloženého mezi jednoduchými nebo dvojitými uvozovkami

Práce s regulárními výrazy je v programovacím jazyce Java řešena skrze vzory (Pattern), které je vhodné před samotným použitím zkompilovat. Vstupem metody *compile*, která je volána přímo na objektu typu Pattern, je textový řetězec. V případě tohoto regulárního výrazu, jej bylo nutné přepsat, podle definovaných pravidel. Speciální znaky jako jsou zpětné lomítko, horní dvojité i jednoduché uvozovky, se musí zapisovat s jedním znakem zpětného lomítka navíc.

## 5.3 Třída LinkInformator

Objekt definovaný na této třídě má informativní charakter. Aktualizuje stav odkazů v tabulce v hlavním okně, počítá čas, po který aplikace je spuštěna a aktualizuje počet zpracovaných odkazů. Tento objekt je po vykonání práce uspán na 100 ms a po uplynutí tohoto času je automaticky probuzen k vykonání té samé činnosti.

Objekt typu *LinkInformator* běží po celou dobu zpracovávání odkazů. Je ukončen *LinkStorageProcesorem* po zkontrolování všech odkazů.

# 6 Grafické uživatelské prostředí

Všechny grafické prvky aplikace jsou obsaženy v balíčku *frames*. V této kapitole jsou popsána všechna okna, se kterými se může uživatel setkat při běhu aplikace

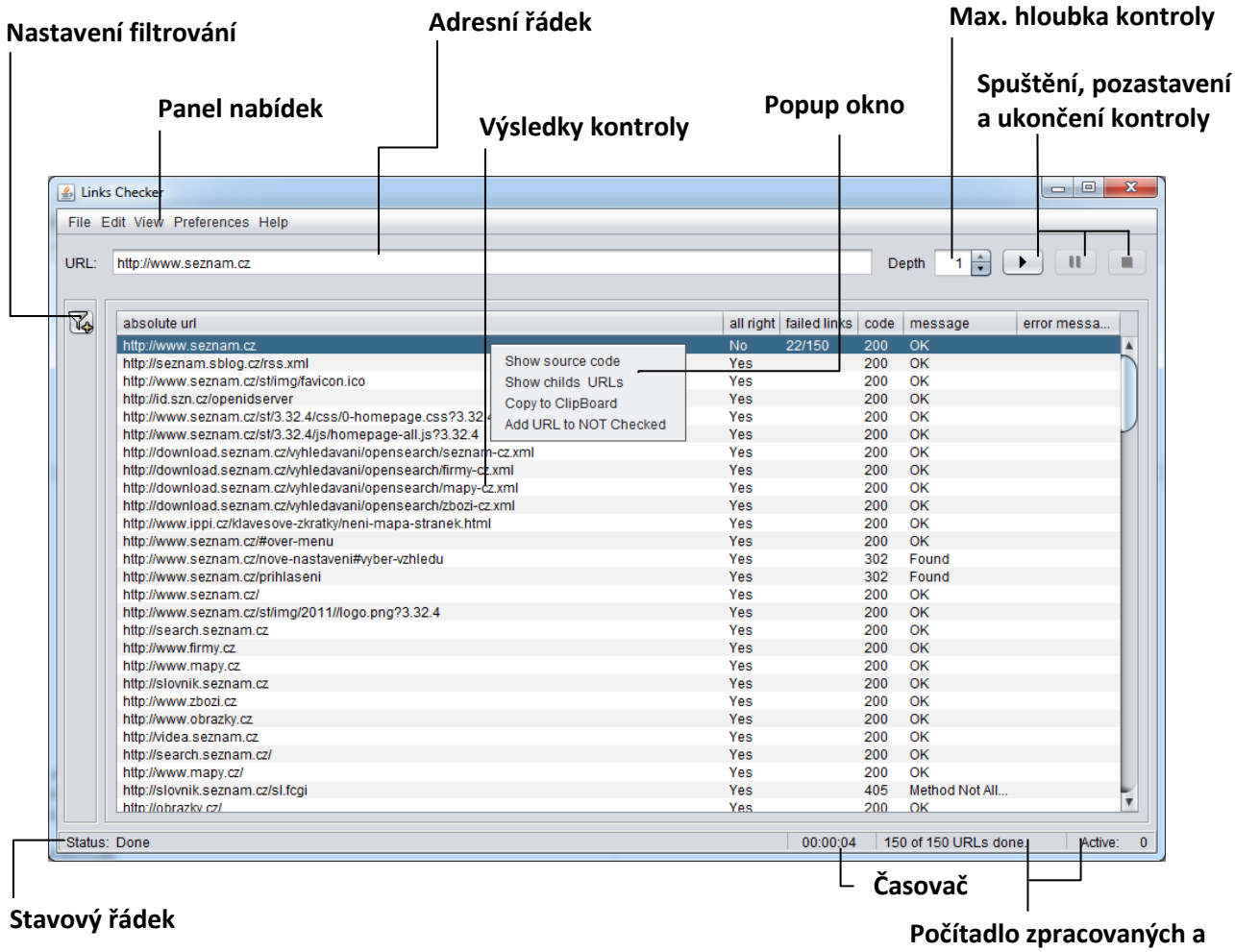
## 6.1 Web Links Checker

Aplikaci Web Links Checker lze spustit na každém počítači s nainstalovaným prostředím pro běh aplikací Java Runtime Environment verze 7. Spuštění aplikace se provede dvojklikem myši na zdrojový soubor s názvem "LinksChecker.jar" nebo přes příkazový řádek pomocí příkazu:

```
java -jar "LinksChecker.jar"
```

Po spuštění aplikace se zobrazí okno, s několika hlavními prvky:

- panel nabídek – umožnuje přístup k nejdůležitějším příkazům aplikace
- adresní řádek – zadávání URL adres
- hloubka kontroly – upřesňuje do jaké úrovně se má daná webová stránka otestovat
- tlačítka pro spuštění, pozastavení a zastavení kontroly
- tlačítko pro vyvolání okna filtrování
- stavový řádek – do tohoto řádku se vypisují informační údaje, jako stav aplikace, počet zkontrolovaných odkazů, počet souběžně běžících vláken, a uplynulý čas kontroly.
- tabulka pro výpis výsledků kontroly – zde jsou vypisovány všechny nalezené odkazy




**Obrázek 14: Hlavní okno aplikace a její části**

## 6.2 Panel nabídek

Panel nabídek obsahuje všechny hlavní komponenty pro ovládání aplikace. Položky v této nabídce jsou rozděleny do 5 kategorií – File, Edit, View, Preferences a Help. Každá tato nabídka má několik dalších položek.

- menu ***File***
  - Check New (Ctrl + N)

    Umožňuje spustit novou kontrolu s vymazáním výsledků předchozí kontroly. Tento příkaz je možné spustit klávesovou zkratkou Ctrl + N.
  - Export to CSV file (Ctrl + S)

    Možnost exportu filtrovaných odkazů do souboru typu CSV. Defaultně jsou hodnoty odděleny středníkem. V předvolbách aplikace na záložce

Export lze tento oddělovač změnit. Nadefinovaná zkratka pro tento příkaz je Ctrl + S.

- Exit (Alt + F4)

  Vypnutí aplikace.

- menu ***Edit***
  - Copy (Ctrl + C)

    Zkopíruje do schránky označený odkaz i s dalšími hodnotami, které jsou v tu danou chvíli zobrazeny v hlavním okně tabulky.
  - Find (Ctrl + F)

    Umožňuje vyhledávat v zobrazených výsledcích.
  - Find Next (F3)

    Pokračuje ve vyhledávání.
- menu ***View***
  - Change view to basic/advanced

    Zobrazení Basic – základní zjednodušené zobrazení, při kterém jsou pevně dány zobrazené sloupce a po dokončení kontroly jsou vygenerovány křížové reference mezi odkazy, tzn. možnost zobrazení počtu chybných odkazů na jednotlivých zkontrolovaných adresách a zobrazení těchto odkazů. Oproti tomu zobrazení Advanced – rozšířené zobrazení, vypíše všechny adresy do tabulky s uživatelem nadefinovanými zobrazenými sloupci.
  - Resize All Columns

    Změní šířku všech sloupců podle nejdelší hodnoty, která se nalézá v daném sloupci.
- menu ***Preferences***

  Umožňuje nastavit např. počet vláken zpracovávající odkazy, kontrolu odkazů s nestejným jménem hostitele, výběr zobrazovaných sloupců v tabulce pro zobrazení Advanced View, či možnost zájmu/nezájmu o konkrétní odkazy nebo skupiny odkazů

- menu ***Help***
  - About

    Informace o verzi aplikace a jejím autorovi.

## 6.3 Panel s výsledky

Největší část okna zabírá panel se zobrazenými výsledky, ve kterém se nacházejí všechny filtrované odkazy. Defaultně je nastaveno zobrazování odkazů s jiným stavovým kódem, než je kód 200 (OK) a odkazy, které nejsou podporovány nebo ještě nejsou otestovány. V tomto panelu je možné na kteroukoliv položku kliknout. Ať už se jedná o výsledek nebo záhlaví samotného panelu s výsledky.

Po kliknutí na určitý sloupec záhlaví panelu je možné zobrazené výsledky třídit, a to sestupně nebo vzestupně. V případě dvojitého kliknutí levým tlačítkem myši na oddělovač mezi sloupci lze sloupec rozšířit nebo naopak zúžit podle nejdelší hodnoty v daném sloupci. Dalšími možnostmi v tomto panelu, je označení jedné nebo více hodnot, které je možné přes položku v menu File vyexportovat do CSV souboru. Každý odkaz v panelu s výsledky umožňuje vyvolat pravým tlačítkem myši kontextové menu, přes které lze:

- zobrazit zdrojový kód dané URL adresy
- zobrazit všechny potomky dané URL adresy
- zkopírovat odkaz do schránky
- přidat / odebrat daný odkaz do seznamu nekontrolovaných adres

## 6.4 Předvolby (Preferences)

Před samotným spuštěním kontroly může uživatel změnit nastavení programu.

Nastavení předvoleb je rozděleno do 3 skupin (záložek):

- General – Obecné nastavení
  Na této záložce může uživatel nastavit počet simultánně běžících vláken, volba kontroly odkazů s jiným jménem hostitele, nežli byl zadán do adresního řádku a seznam adres, které se mají kontrolovat a adres nebo naopak ignorovat.

  Simultánní počet vláken je v základním stavu nastaven na hodnotu 20. V případě zrychlení kontroly je možné nastavit počet vláken na vyšší hodnotu. Zde ale opatrně. Webový server by mohl chybně detekovat velký počet dotazů jako robota útočícího na systém a následně zablokovat komunikaci se serverem na několik minut nebo i déle.

Kontrola odkazů s jiným jménem hostitele, nežli byl zadán do adresního řádku, umožňuje kontrolovat např. webovou stránku *http://www.abc.cz* a pokud na této stránce bude nalezen odkaz s jiným jménem hostitele, např. *http://www.def.cz*, tak se dotyčná adresa kontrolovat nebude.

Textová pole „Kontroluj odkazy začínající na:“ a „Nekontroluj odkazy začínající na:“ poskytují uživateli volbu konkrétního seznamu adres, kterými se má program zabývat, nebo naopak které má vynechat.

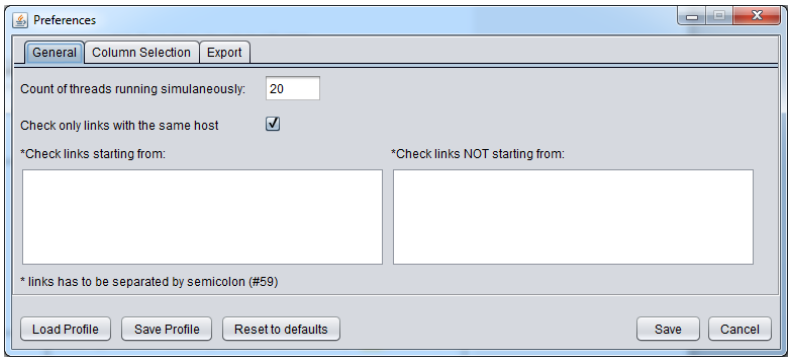
- Column Selection – Výběr sloupců

  Výběr sloupců umožňuje uživateli vybrat zobrazované sloupce v tabulce na hlavním panelu pro zobrazení Advanced View. Defaultně jsou zobrazovány sloupce absolutní adresy odkazu, stavového kódu, stavové zprávy a chybového hlášení. V případě, že uživatele zajímá například původní nalezená url adresa, netransformovaná na absolutní adresu, je potřeba v nabídce předvoleb zaškrtnout zobrazování sloupce url. Další zobrazovanou možností je například typ tagu, ve kterém byl odkaz nalezen a atribut tohoto tagu. Zobrazovat lze ještě úroveň stránky, na které byl odkaz nalezen, jméno hostitele, pozici v řádku a sloupci, popř. adresu serveru, na který byl odkaz přesměrován.

- Export

  Zde je možné změnit oddělovač hodnot při exportování odkazů do CSV souboru nebo uzavření hodnot exportovaného textu do uvozovek. Oddělovač je defaultně nastaven na hodnotu středníku.

V případě, že bude uživatel pracovat na více projektech najednou, lze uložit použité nastavení pro konkrétní projekt a později ho nahrát. Defaultní jméno souboru nastavení předvoleb je DefaultProfile.ini.




**Obrázek 15: Okno nastavení předvoleb**

## 6.5 Spuštění testování

Před spuštěním testování je potřeba zadat URL adresu do adresního řádku a nastavit požadovanou hloubku kontroly. Adresa, kterou uživatel zadává, by měla být v následujícím formátu:

[protokol]://[doména 3. úrovně].[doména 2. úrovně].[nejvyšší doména]

V případě, že chybí protokol nebo doména 3. úrovně, tak je adresa brána jako validní. Pokud, ale zadaná URL adresa neobsahuje ani protokol, ani doménu 3. úrovně je brána jako nevyhovující.

Po zadání adresy, už stačí jenom spustit kontrolu a to se provede, tlačítkem Start. Průběh samotné kontroly je možné sledovat v okně aplikace.

V dolní části okna je vidět celkový počet dosud nalezených souborů a kolik jich program v danou chvíli již prověřil. Test je možné kdykoliv pozastavit, znovu spustit, nebo zcela zastavit tlačítky na nástrojové liště (fungují obdobně, jako tlačítka v multimediálních přehrávačích).

## 6.6 Filtrování odkazů

Nalezené odkazy lze filtrovat podle několika způsobů:

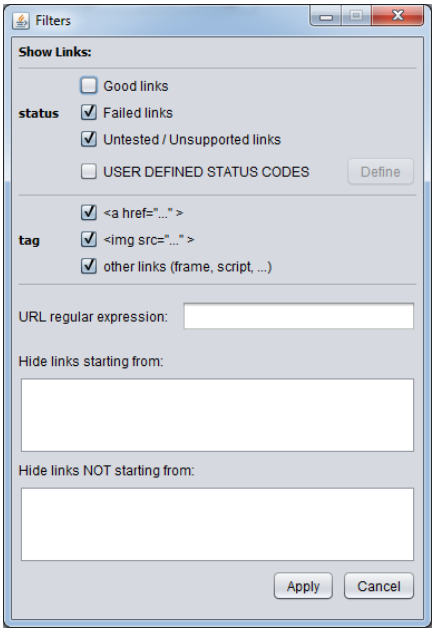
- stavu odkazu
  Filtrování podle tohoto typu lze rozdělit na odkazy, které jsou v pořádku, tzn. vracející stavový kód 200, odkazy které vracejí jiný kód a odkazy, které ještě nebyly doposud testovány. Poslední volba filtrování je podle uživatelsky definovaného stavového kódu

- HTML tagu
  Tento způsob filtrování umožňuje zobrazit odkazy nalezené v tagu odkazu, obrázku nebo ve všech ostatních typech tagů.

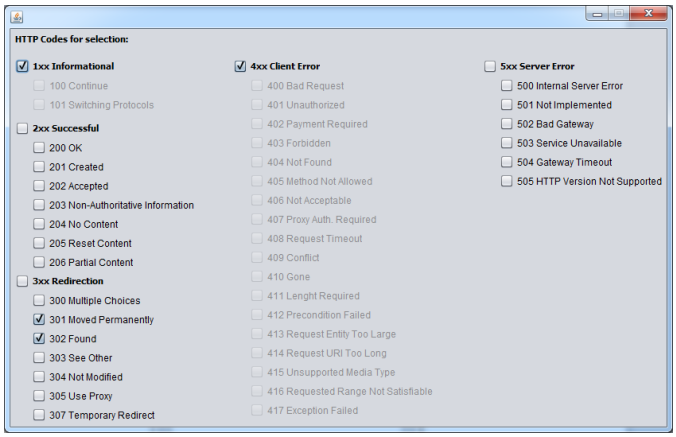
- adresy odkazu začínajícího nebo nezačínajícího zvolených prefixem

- pomocí regulárního výrazu

  Toto filtrování umožňuje filtrovat odkazy podle hledaného řetězce znaků odpovídající určitému vzoru.




**Obrázek 16: Okno nastavení filtrů**



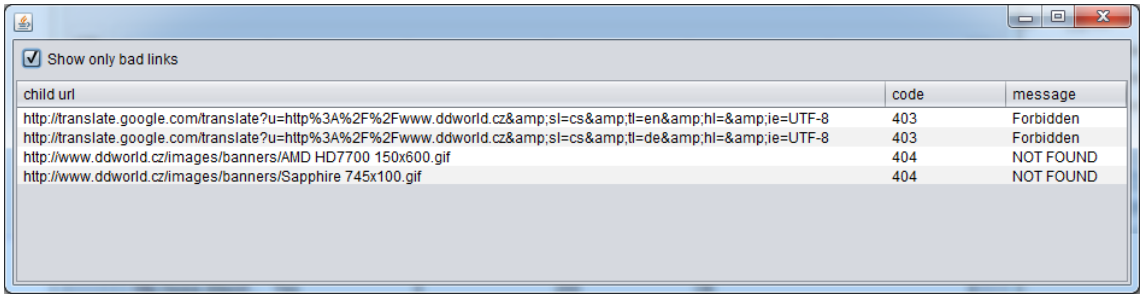

**Obrázek 17: Okno filtrování podle uživatelsky definovaných kódů**

V případě volby uživatelsky definovaných zobrazených stavových kódů jsou ostatní možnosti zobrazení podle stavu odkazu deaktivovány. Tlačítko pro definici kódů se aktivuje a uživatel si může zvolit libovolné návratové kódy. Po zvolení všech kritérií filtrování, se změny aplikují stiskem tlačítka Apply.

## 6.7 Chybné odkazy

Zobrazení seznamu chybných odkazů je možné dvěma způsoby.

První způsob je cestou filtrování odkazů a jejich vypsání přímo do hlavního okna aplikace. Druhý způsob je možný pouze při základním (zjednodušeném) zobrazení aplikace. V základním režimu zobrazení aplikace totiž po dokončení kontroly, dochází k vytvoření křížových referencí mezi odkazy a spočítání všech odkazů (potomků), jejichž návratový kód je jiný než 200(OK) pro všechny odkazy, které se vyskytly na nějaké konkrétní testované stránce.




**Obrázek 18: Zobrazení chybných odkazů v jednom HTML dokumentu**

# Závěr

Dnešní podobu informačních technologií si bez možnosti připojení k internetu a přenosu a sdílení dat neumíme vůbec představit. Ještě před pár lety byly tyto technologie drahé a nepřístupné široké laické veřejnosti. Lidská společnost tuto technickou vymoženost bere dnes jako samozřejmou a snaží se jí dle svých schopností využívat.

V současné době jsou v internetu, tedy v síti sítí, připojeny tisíce serverů a počítačů, na kterých jsou miliony navzájem odkazy provázaných stránek s informacemi. Jednou ze služeb, které tato síť, propojující počítače na celém světě nabízí, je World Wide Web (www). Tato služba umožňuje přenos hypertextových dokumentů a funguje na principu klient a server. V praxi to znamená, že dokumenty jsou uloženy na serverech a ostatní počítače posílají serverům požadavky na dokumenty, a servery jim požadované informace zasílají zpět. Z tohoto důvodu je potřeba, aby odkazy na webových stránkách byly funkční a odkazovaly na požadované informace, které se vyhledávají.

V této diplomové práci je zmapována problematika odkazů na webových stránkách, kdy se nejprve věnoval základním aspektům webových stránek, a to protokolu pro výměnu hypertextových dokumentů, typům adres na webových stránkách včetně nejpoužívanějších typů schémat.

Byla provedena rešerše současných a dostupných nástrojů pro kontrolu webových odkazů. V praktické části se především věnuje vlastnímu návrhu programu pro kontrolu odkazů webových stránek, kde je velká pozornost věnována zpracování výsledků a následnému zobrazení chybných odkazů.

Navržený program se nemusí instalovat, ale je spustitelný na libovolném počítači. Je zde však jedna podmínka a to, že musí být na počítači nainstalována Java (Java Runtime environment, JRE). V navržené aplikaci byl kladen velký důraz ze strany autora především na grafické uživatelské prostředí. Program je po grafické stránce velmi uživatelsky přívětivý a práce s ním je velmi snadná a intuitivní. Nabízí dvě úrovně zobrazení - jednoduché, kde se zobrazí pouze seznam špatných odkazů zabalených pod jediný odkaz a to pod absolutní adresu stránky, na které byly tyto odkazy nalezeny. Pro zobrazení neúspěšně otestovaných odkazů pak slouží jednoduchý dvojklik myší na danou adresu stránky, ze které chceme tyto odkazy zobrazit.

Dále je zde zobrazení normální, kde jsou odkazy uvedeny do tabulky, tak jak byly nalezeny na stránkách včetně vypsání transformovaných relativních adres na adresy absolutní, typu kontrolovaného atributu, návratového kódu a jemu náležející návratové zprávě.

Navržený program se některými částmi podobá již exitujícím nástrojům pro kontrolu webových odkazů a to ať už komerčním nebo volně šiřitelným. Takovými jsou například předvolby programu, kde je možné zvolit počet procesů běžících simultánně, maximální hloubku kontrolované úrovně, možnost kontrolování či nekontrolování odkazů začínajících daným řetězcem. Dále možnosti filtrování odkazů podle správně či neúspěšně otestovaných odkazů nebo podle uživatelem konkrétně definovaných návratových kódů. Navíc navržený program umí oproti konkurenčním aplikacím filtrovat odkazy pomocí regulárních výrazů a ještě podle typu tagu ukrývajícího cíl odkazu. Samozřejmostí je možnost uložení všech předvoleb to souboru a jejich načtení při dalším spuštění.

Do budoucna by bylo možné aplikaci rozšířit o možnost generování stromové struktury nalezených odkazů či doplnění výzvy pro autorizaci uživatele v případě zabezpečených webových stránek.

Autor si přeje, aby vytvořená aplikace, byla přínosem a užitečným nástrojem pro programátory webových stránek a pomohla odhalit nefunkční či špatně zapsané hypertextové odkazy na vytvářených stránkách.

# LITERATURA


[1] KOSEK, Jiří. *HTML: tvorba dokonalých www stránek : podrobný průvodce*. Vyd. 1. Praha: Grada, 1998, 291 s. ISBN 80-716-9608-0.

[2] RAGGETT, Dave. *Raggett on HTML 4* [online]. Harlow Reading Menlo Park : Addison-Wesley, 1998. 437 s. ISBN 0-201-17805-2. URL: <http://www.w3.org/People/Raggett/book4/>

[3] SKLENÁK, Vilém. *Data, informace, znalosti a Internet*. Vyd. 1. Praha: C. H. Beck, 2001. 507 s. ISBN 80-7179-409-0.

[4] AULDS, Charles. *Linux : administrace serveru Apache*. Vyd. 1. Praha: Grada, 2003. 535 s. ISBN 80-247-0640-7.

[5] RFC 2616. *Hypertext Transfer Protocol -- HTTP/1.1* [online]. [cit. 2012-05-09]. URL: <http://tools.ietf.org/html/rfc2616>

[6] RFC 3986. *Uniform Resource Identifier (URI): Generic Syntax* [online]. [cit. 2012-06-17]. URL: <http://tools.ietf.org/html/rfc3986>

[7] RFC 1738. *Uniform Resource Locators (URL)* [online]. [cit. 2012-07-07]. URL: <http://tools.ietf.org/html/rfc1738>

[8] PÍSEK, Slavoj. *HTML : tvorba jednoduchých internetových stránek*. Vyd. 2. Praha: Grada, 2006. 108 s. ISBN 80-247-1767-0.

[9] DARWIN, Ian F.. *Java : kuchařka programátora : [vzory a řešení pro vaše aplikace]*. Vyd. 1. Brno: Computer Press, 2006. 798 s. ISBN 80-251-0944-5.

[10] SPELL, Brett. *Java : programujeme profesionálně*. Vyd. 1. Praha: Computer Press, 2002. 1022 s. ISBN 80-7226-667-5.

[11] GOYVAERTS, Jan. *Regular expressions cookbook.* Vyd. 1. Beijing: Oreilly, 2009. 493 s. ISBN 978-0-596-52068-7.

# PŘÍLOHA A

## Třída LinkStorageProcessor

Tento zdrojový kód představuje hlavní část (třídu) programu, v které se provádí volání metody pro načítání obsahu zdrojového kódu kontrolovaných odkazů, včetně udržování maximálního počtu uživatelem definovaných současně běžících vláken.

```
/*
 V této třídě se pomocí volání třídy LinkReader načítá zdrojový kód
 kontrolovaných odkazů a zároveň udržuje maximální počet současně běžících
 vláken nadefinovaných v třídě PreferencesDataObjects.
*/
package process;

import data_objects.CheckedLink;
import data_objects.Constants;
import data_objects.LinkStorage;
import data_objects.PreferencesDO;
import frames.NewJFrame;
import helpers.InputLinkChecker;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import javax.swing.JOptionPane;


public class LinkStorageProcessor implements Runnable{
    private static List<LinkReader> runningLinkReaders = new
         ArrayList<LinkReader>();

    private String url;
    private NewJFrame mainApp;
    /*
     konstruktor třídy se 2 parametry: vstupní url adresa a objekt typu
     NewJFrame, který umožňuje přístup k hlavnímu oknu a tím umožňuje měnit
     popisky informativních objektů - počet běžících vláken, čas a stav práce
    */
    public LinkStorageProcessor(String url, NewJFrame mainApp) {
        this.url = url;
        this.mainApp = mainApp;
    }

    @Override
    public void run() {
         //kontrola vstupní url adresy, zdali je ve správném formátu
         if (InputLinkChecker.isCorrect(url)) {

             //transformace vstupní url adresy
             String transformedURL = InputLinkChecker.getTransformedUrl(url);

             /*
              vytvoření objektu odkazu pro první (vstupní odkaz) a nastavení
              atributů (id kontrolovaného odkazu, hloubka úrovně, na které se
              nachází, vstupní a transformované url adresy)
             /*
             CheckedLink root = new CheckedLink();
             root.setId(LinkStorage.getNextId());
             root.setLevel(0);
             root.setUrl(url);
             root.setMessage(Constants.TEXT_NOT_CHECKED);
             root.setTransformedUrl(transformedURL);
```

```
        /*
         Pokud je zavolena kontrola odkazů pouze se stejnou doménou, je do
         proměnné absoluteHost ve třídě PreferenceDataObjects nastavena
         hodnota domény ze vstupní url adresy.
        */
        if (PreferencesDO.onlyAbsoluteHost) {
            URL rootUrl;
            try {
                rootUrl = new URL(transformedURL);
                PreferencesDO.absoluteHost = rootUrl.getHost();
            } catch (MalformedURLException e) {
                System.out.println("Error to check root link." + e);
                root.setErrorMessage("Error while getting host name.");
            }
        }
        /*
         Vytvoření objektu typu LinkReader, ve kterém se provádí kontrola
         a načtení celého obsahu zdrojového kódu, včetně vyhledávání
         nových odkazů – všechny nalezené odkazy jsou uloženy v objektu
         LinkStorage do listu nezkontrolovaných odkazů.
        */
        LinkReader mainReader = new LinkReader(root);
        LinkStorageProcessor.addRunningLinkReader(mainReader);
        /*
         přidání vstupního url do objektu typu LinkStorage(tzv. úložiště
         odkazů)
        */
        LinkStorage.addCheckedLink(root);
        //spuštění prvního objektu LinkReader
        mainReader.start();


/*
Dokud v objektu LinkStorage v listu nezkontrolovaných odkazů existují
odkazy nebo dokud jsou aktivní některé objekty (vlákna) kontrolující
odkazy, pak jsou vytvářeny nové objekty (vlákna) do maximálního počtu
uživatelem stanoveného limitu.
*/
while (LinkStorage.hasUncheckedLinks() || runningLinkReaders.size()> 0) {
    if (runningLinkReaders.size() < PreferencesDO.threadCount) {
         int countForNewThreads =
             PreferencesDO.threadCount - runningLinkReaders.size();
         for (int i = 0; i < countForNewThreads; i++) {
             if (LinkStorage.hasUncheckedLinks()) {
                CheckedLink cl = LinkStorage.getOneUncheckedLink();
                LinkReader reader = new LinkReader(cl);
                LinkStorageProcessor.addRunningLinkReader(reader);
                LinkStorage.moveUnToCheckedLink(cl.getTransformedUrl());
                reader.start();
             }
         }
    }
}

/*
Po ukončení kontroly odkazů, je zastaveno vlákno, informující o stavu
kontrolovaných odkazů, tlačítka pro pozastavení a ukončení kontroly jsou
uvedeny do základního stavu a stav kontroly je nastaven na „Done“.
*/
LinkInformator.setRun(false);
mainApp.timeCounter.cancel();
mainApp.resetButtons();
mainApp.jLabelState.setText(Constants.STATUS_DONE);
helpers.TableHelpers.enableTableSort(mainApp.jTableResults);

JOptionPane.showMessageDialog(mainApp, "Done. Checked " +
    LinkStorage.getCheckedLinks().size() + " links in " +
    mainApp.jLabelTime.getText() + ".");
```

```
        /*
        V případě, že je vstupní url adresa ve špatném tvaru, je uživatel
        informován a aplikace je uvedena do zákl. stavu
        */
        } else {
            JOptionPane.showMessageDialog(null, Constants.URL_BAD_FORMAT, null,
               JOptionPane.WARNING_MESSAGE);
            mainApp.resetAllMainWindow();
        }
    }

    //metoda odstraňující kontrolující vlákno ze seznamu aktivních vláken
    public static synchronized void removeRunningLinkReader(LinkReader
        linkReader) {
          runningLinkReaders.remove(linkReader);
    }

    //metoda přidávající kontrolující vlákno do seznamu aktivních vláken
    private static synchronized void addRunningLinkReader(LinkReader
        runningThread) {
          runningLinkReaders.add(runningThread);
    }

    //metoda vracející seznam kontrolujících vláken
    public static synchronized List<LinkReader> getRunningLinkReaders() {
          return runningLinkReaders;
    }

    //metoda vracející počet kontrolujících vláken v seznamu aktivních vláken
    public static synchronized int getRunningLinkReadersCount() {
          return runningLinkReaders.size();
    }

    //metoda odstraňující všechny vlákna ze seznamu aktivních vláken
    public static synchronized void clearRunningLinkReaders() {
         runningLinkReaders.clear();
    }

}
```

# PŘÍLOHA B

## Obsah CD

| | |
|---|---|
| /app/src/ | Složka se zdrojovým kódem aplikace |
| /app/dist/ | Složka se zkompilovanou aplikací |
| /jre/ | Složka se staženým Java Runtime Environment 7 |
| /WebLinksChecker.pdf | Text práce v PDF |